SPATIO-TEMPORAL DATA INTERPOLATION FOR DYNAMIC SCENE ANALYSIS

A Thesis
Presented to
The Academic Faculty

by

Kihwan Kim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
May 2012

SPATIO-TEMPORAL DATA INTERPOLATION FOR DYNAMIC SCENE ANALYSIS

Approved by:

Professor Irfan Essa, Advisor
School of Interactive Computing
Georgia Institute of Technology

Professor Greg Turk
School of Interactive Computing
Georgia Institute of Technology

Professor Thad Starner
School of Interactive Computing
Georgia Institute of Technology

Professor Jessica K. Hodgins
Robotics Institute
Carnegie Mellon University and Disney Research Pittsburgh

Professor James M. Rehg
School of Interactive Computing
Georgia Institute of Technology

Date Approved: 6 December 2011

To the memory of my father, Kwangwoong Kim,

To my mother, Changok Song.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Irfan Essa for his support of my Ph.D. study and research. He gave me the confidence to explore my research interests and the guidance to avoid getting lost in my exploration.

Beside my advisor, I would like to thank the rest of my thesis committee members: Dr. James M. Rehg, Dr. Greg Turk, Dr. Thad Starner, and Dr. Jessica K. Hodgins for their encouragement and insightful comments. My sincere thank also goes to Dr. Ariel Shamir and Dr. Iain Matthews who gave me countless motivations, intuitions, and advices at Disney Research.

In particular, I am very grateful to Dongryeol Lee, my friend, for his help and advice during all the time in my Ph.D. study. I am also grateful to my colleagues Matthias Grundmann, Sangmin Oh, Byungmoon Kim, Dongshin Kim, Matt Flagg, Jeil Choi, Tucker Hermans, Denis Ayleshin, Vinay Bettadapura, Yuting Ye, Grant Schindler, Takaaki Siratori, Sehoon Ha, Kang Lee and many friends in computational perception laboratory (CPL) and graphics laboratory in Georgia Institute of Technology, and Disney Research Pittsburgh.

Finally, I would like to express my sincerest gratitude to my wife, Yijin Yun for her patience, understanding and support. I could not have done this without her.

TABLE OF CONTENTS

DEDICATION . . . . . **iii**

ACKNOWLEDGEMENTS . . . . . **iv**

LIST OF TABLES . . . . . **viii**

LIST OF FIGURES . . . . . **ix**

I ABSTRACT **1**

II INTRODUCTION **3**

2.1 Thesis Statement 6

2.2 Outline of the Remaining Chapters 6

III RELATED WORK **8**

3.1 Methods for Dynamic Scene Analysis 8

3.1.1 Foreground segmentation 8

3.1.2 Tracking Objects in a Scene 9

3.1.3 Estimating Scene Geometry 10

3.1.4 Motion and Flow Field Analysis 10

3.1.5 Behavioral Models 11

3.2 Scene Visualization techniques 11

3.3 Interpolation and Scattered Data Approximation 13

IV SCATTERED DATA INTERPOLATION AND APPROXIMATION 15

4.1 Radial Basis Function (RBF) Interpolation 16

4.2 Gaussian Process Regression (GPR) (Kriging) 18

4.2.1 Gaussian Process and Data Prediction 19

4.2.2 Gaussian Process Regression and Radial Basis Function . . . 20

4.3 Spatio-Temporal Scattered Data Interpolation and Approximation . 22

V DYNAMIC SCENE VISUALIZATION FROM SPARSELY DISTRIBUTED VIDEOS **24**

5.1 Direct visualization using fixed cameras 25

5.2 Overlapping Cameras, Complex Motions: Sports 27
5.3 Sparse Cameras with Simple Motion: Traffic 32
5.4 Sparse Cameras with Complex Motion: Clouds 37
5.5 Discussion of Experiments and Results 40
5.6 Summary 44

VI GLOBAL MOTION ANALYSIS FOR ADJUSTING THE VIEW POINTS OF A CAMERA 48

6.1 Observation on complex dynamic sports scenes 48
6.2 Motion Field Construction from Video 50
6.2.1 Extracting Individual Ground-Level Motion 52
6.2.2 Dense Motion Field Construction using Spatio-temporal Radial Basis interpolation 55
6.2.3 Detecting Points of Convergence 57
6.3 Results and Evaluation 60
6.3.1 Evaluation of Ground-level Motion Extraction 61
6.3.2 Evaluation of Motion Field and POC detection 62
6.4 Summary 75

VII MOTION PATTERN RECOGNITION AND ANOMALY DETECTION FROM SPATIO-TEMPORALLY SPARSE MOTION TRAJECTORIES 77

7.1 Stochastic Motion Representation to Motion Pattern Recognition . 77
7.2 Gaussian Process Regression for Flow 80
7.2.1 Constructing Posterior Density 80
7.2.2 Mean Flow and Confidence Band 81
7.3 Learning Stable GPRFs 81
7.3.1 Normalization 82
7.3.2 Number of Samples and Variances 83
7.4 Similarity Measurement 85
7.4.1 Measuring the Local Likelihood of Testing Data 85
7.5 Classification and Prediction of Trajectories 86

7.5.1 Similarity for Complete Trajectories 86
7.5.2 Prediction from Incomplete Trajectory 87
7.6 Anomaly Detection 88
7.7 Results and Evaluation 90
7.8 Summary 94

VIII STOCHASTIC APPROACH FOR GLOBAL MOTION ANALYSIS IN DYNAMIC SCENES WITH CAMERA MOTION 97

8.1 Detecting Regions of Interest 99
8.1.1 Computing Motion on the Ground 99
8.1.2 Stochastic Motion Field 100
8.1.3 Detecting Locations of Convergence 102
8.1.4 Measuring the Similarity with Field of View 104
8.2 Evaluation and Results 105
8.2.1 Similarity with actual camera operator's view 106
8.2.2 Computational Expense 108
8.2.3 Qualitative evaluations for both moving and static cameras . 109
8.3 Conclusion 110

IX SUMMARY AND CONCLUSION 113

X PAPERS RELATED TO THIS THESIS BY THE AUTHOR . . 116

REFERENCES 117

VITA 129

LIST OF TABLES

1 **Categorization of methods used for scene visualization** 12

2 **Detection errors on the number of cameras** 61

3 **Average errors of minimum distances** 75

4 **Computational time with various sizes of top-view** 75

5 **Dataset description** 92

6 **Trajectory recognition using complete trajectories** 92

LIST OF FIGURES

1 **Example of a simple scattered data interpolation:** (a) observed scattered data points (shown in blue) regarding geometric properties (e.g., the surface) and (b) a prediction of the values at unknown locations. 16

2 **One-dimensional example of a scattered data interpolation using thin-plate RBF and GPR:** The red points in (a) and (b) denote input data points (x axis) and their values (y axis), the black curves indicate the interpolated/approximated values (a) interpolated values using thin-plate RBF; keeping C^1 continuity as a geometric constraint (b) approximated values (mean) from GPR, and (c) a color-bar in the right side represents a certainty level (values indicate the band size in a Gaussian distribution of 95 percent based on its variance); red indicates a higher certainty (small variance); and blue indicates a lower certainty (large variance) 21

3 **Abstracts of spatio-temporal SDI/SDA in a dynamic scene domain** Examples describe how SDI/SDA is applied to dynamic scene analysis. For each image, each rectangle indicates a frame in a video. Blue dots indicate all types of features, including detected/tracked objects in video frames. Green dotted arrows show a velocity vector based on the corresponding features between frames. Red arrows represent interpolated motions: (a) the generation of a continuous dense vector field in a frame at time t using observations from sparse motion data collected from previous frames, (b) a spatio-temporal dense field describing motion tendencies over a specific duration, and (c) propagated dense motion information even in unobserved locations 22

4 **An overview of the "Augmented Earth Map":** We make use of 36 videos to add dynamic information to city visualization. Dynamism in each input video of traffic, people, and clouds is extracted and then mapped onto the Earth map in real-time. Each video patch in the figure represents an input source. 25

5 **Example of Direct Mapping:** (a) Pedestrians are tracked from single video (b)Tracked pedestrians are matched to motion capture data, then mapped onto virtual plane space, then rendered only within a coverage of the given field of view. 27

6 **The relationship between each view:**(a) Virtual view $\mathbf{f}_v$, Sample view $\mathbf{f}_i$ and the angle between them θ (b) Two consecutive views $\mathbf{f}_i$, $\mathbf{f}_{i+1}$, Back-projected virtual view $\mathbf{f}_v$ and blended rectified view $\hat{\mathbf{f}}_v$. Note that θ_i is an angle between $\mathbf{f}_i$ and $\mathbf{f}_v$ and θ_{i+1} is an angle between $\mathbf{f}_{i+1}$ and $\mathbf{f}_v$, (c) Source video having view $\mathbf{f}_i$(d) Back-projected virtual view $\mathbf{f}_v$ (e) Rectified view $\hat{\mathbf{f}}_v$: Note that (c) and (d) look almost similar view since the θ is almost zero. 28

7 **The range of approximately invariant to distortion:** (a) and (b) both are back-projected scenes from same video (Notice that within small change of viewing angle it still looks reasonable). 29

8 **View blending with different interpolations**, where ω_i is 0.6 :(a) Linear (b) Bicubic (c) Bicubic and Background blending: f is $f(\omega_i)$, g is $g(\omega_{i+1})$ and h is ω_{bkg} which is a weight for subtracted background image. (d),(e) and (f) show the back-projected views based on each interpolation. The gray arrows in (c) show the ranges of ω where the distortion is minimized, as shown in Fig. 7(a)(b). 30

9 **Graph-based representation of traffic nodes:** Red nodes indicate observable region $\mathbf{M}(\mathbf{X})$ and Green nodes are unobserved regions $\tilde{\mathbf{M}}(\mathbf{X})$. (a) The middle chain corresponds to the traffic conditions on the graph which represents the traffic system. Node index i is 0 to 2 in this example (b) split: outgoing regions $\mathbf{O}(\mathbf{X_i})$ from node X_i are marked. (c) merging: incoming regions $\mathbf{I}(\mathbf{X_i})$ to node X_i are marked. 34

10 **Simulated cars:** (a) Each object(car) acts as an autonomous agent controlled by (1) local reaction based on their own perception range, (2) global dynamics extracted from video (b) Cars' reaction when average speed of upper-left corner is forced to be very slow, while other corners are controlled by real data. 34

11 **Generating clouds layers procedurally using videos:** (a) 4 input videos (Green region is an unobserved region)(b) Globally interpolated density map from RBF (c) result cloud layer (d) registered onto the sky dome in Earth Map environment. 38

12 **Various results of sky rendering based on multiple videos at different times.** 40

13 **Results from our prototype system using 36 videos: 1. OCCM (View Blending):** (a) 5 Cameras for soccer game (b) Two broadcasting footages of NCAA Football game (c) Three surveillance cameras. **2. SCSM (Traffic):** (d) Merging Lanes (e) The steering behaviors of cars are demonstrated in the slightly curved way (f) 8 cameras for larger scale traffic simulation including merge and split (g) Rendered traffic Scene and corresponding simulated scene **3. DM (Pedestrians):** (h) Direct mapping of pedestrian having simple motion (i) Visualization of pedestrians and cars in the street **4. SCCM (Clouds):** (j) Four videos for clouds and sky generation **5. Customized visualization using user interface:** (k) Traffic flow are visualized in a line map by adjusting the scale of moving objects (l) The game scenes can be augmented into different stadium. Please see the video on the project web site. 41

14 **Qualitative evaluation of traffic in different lighting and weather conditions** Rendered Dynamic scenes based on **(Top)** video data captured at night : Resulting scene demonstrates sparse traffic distribution with high velocity across almost entire nodes, **(Bottom)** video data captured during snowing weather condition : Despite heavy noises in the video due to snow, the result shows flows that realistically demonstrate real-world traffic; one lane (marked as red) shows densely populated heavy traffic while the other lane (yellow) is relatively sparser. 42

15 **Examples of how players movement indicates play evolution in a dynamic sport scene:** The motion field on the ground is denoted as white arrows, and the locations where play evolution is predicted are denoted as red iso-contours. Ball location is highlighted with a circle. **Top:** The goalkeeper passes the ball to the last defender (red box) while an offender (yellow box) is moving to intercept him. **Bottom:** One second (30 frames) later, at the moment of interception (position 1), the goalkeeper and another defender (yello boxes) are moving in the direction of position 2. This indicates the possible location of a future event. 50

16 **Geometric Constraints**: (Upper row) Each view ($\mathbf{I}_k$) has a vertical vanishing points ($\mathbf{v}_k$). (Bottom left) In the top-view warped image ($\mathbf{I}_k^{top}$), players are distorted in the direction of the projected vanishing points ($\hat{\mathbf{v}}_k$) of each view. (Bottom right) The location of a player on the ground is identified by the intersection of these projections on the ground plane. 51

17 **Position Confidence in top-view**: (a) Overlapped top-view of warped views from each angle (each view of a player is warped over the direction of vertical vanishing points)(b) Normalized probability of the summation of the warped foreground probabilities. Note that the region which is close to the ground has higher probability (c) In the presence of shadow. 52

18 **Finding the optimal location using Geometric Constraints**: (a) Samples ($\tilde{\mathbf{x}}_{ij}$) along lines from the evaluating point $\mathbf{x}$ and each projected VVPs $\hat{\mathbf{v}}_k$. Since the number of foreground samples is small, the window moves to top-right. (b) Evaluate Eq. 13 at each point $\mathbf{x}_{ij}$ inside $\mathbf{W}_{init}$. (c) Optimized location. 54

19 **Visualizing the components for evaluation of player location at time t and $t-a$.** The player in the upper row remains in same location, and the one in bottom row moves. Colors range from blue (low) to red (high). In each row: (a) Values of $\mathbf{G}(\mathbf{x})$ in $\mathbf{W}_{init}$, (b) Values of $\mathbf{G}(\mathbf{x})^t$ in $\mathbf{W}_{opt}$ at current frame t, (c) Values of $\mathbf{G}(\mathbf{x})^{t-a}$ in $\mathbf{W}_{opt}$ at previous frame $t-a$, (d) Distances of the color histogram between frame t and $t-a$ within $\mathbf{W}_{opt}$, (e) Weighted sum of $\mathbf{G}(\mathbf{x})^{t-a}$ and $\mathbf{C}(\mathbf{x})^{t-a}$, and (f) A player of the evaluation and sampled colors. We extract velocity vectors between t and $t-a$ by subtraction of minima between (b) and (e). This vectors are shown in (b) as white arrow. . 54

20 **Motion Field** Φ: White arrows represent the dense motion field generated from a sparse set of motions of players movements. Note that for visualization purposes the dense field is displayed sparsely by averaging the flow at each block. 56

21 **Points of Convergence detection**: (a) Starting from the position $\mathbf{x}_{ij}$ and another point having a different magnitude of motion vector as an example. The magnitude of ρ_{ij} is propagated through the point $(i+u_{ij}, j+u_{ij})$. (b) An importance table Ψ is updated by adding propagated confidence along Φ. (c) Pink circles at bottom are the location where the accumulated importance is high enough (larger is higher confidence). (d) Meanshift clustering and Guassian mixture modeling detects two POCs in this case. 58

22 **Divergence and Points of Convergence**: Upper row: Red regions denote the regions where $\nabla\Phi < 0$, and blue regions denote regions where $\nabla\Phi > 0$. Lower row: Φ in (a) has specific singular sink, while Φ in (b) has no specific singular region. In both cases our approach detects the POCs (red elipses). 59

23 **Evaluation of automatic player location on the ground:** We back-project the position onto each view for evaluation. Each R, G and B line with yellow circle denotes the direction of the vertical vanishing points for each view (geometric constraints). 62

24 **Flow generated from simple motions, and their POCs**: Blue circles are the moving objects, and the arrows indicate current velocity. Green circles are the location where each object started from. Blue dots are the trace of each object's past location. Black arrows indicate the block-average of the flow field. Red contours represent the calculated POCs. 63

25 **Example 1 of play evolution.** In all figures, the pink circles are the location of the ball, the arrows denote the initial direction of the movement of the ball, white arrows denote the motion field on the ground, and red contours are the distribution of POC where play evolution aims. In this example - interception & goal keeping. **(Left)** Goalkeeper 1 passes a ball to the last defender 2. A POC appears near the defender, **(Middle)** While the ball is still on the way to the defender 2, another POC appears at different location as offender 3 approaches and the goalkeeper and defender 4 decide to defend. **(Right)** The ball is intercepted by offender 3 and the POC event in the left region actually occurs. Finally, the ball was saved by the goalkeeper. Note that two POCs appear in both possible directions. 64

26 **Example 2 of play evolution - center pass (Left)** Offender 1 dribbles towards the upper corner while offender 2 runs toward the other corner (two POCs). **(Middle)** Offender 1 kicks a center pass. While the ball is travelling the POC near offender 2 becomes larger. **(Right)** Defender 3 intercepts and the ball changes its direction. Offender 2 and another defender approach to the ball. 64

27 **Example 3 of play evolution - back-door and through pass**: **Upper row: (Left)** Offender 1 dribbles forward (a POC is in front of the player). **(Middle)** Offender 1 passes the ball to offender 2 as part of a back-door strategy. A POC appears near offender 2. **(Right)** While the offender 2 waits to receive the ball, offender 1 and the defenders are running toward the goal-post. Another POC appears near goal. **Lower row: (Left)** Before offender 2 kicks the ball, the POC near the goalpost becomes larger. **(Middle)** Offender 2 through-passes to offender 1. **(Right)** Offender 1 attempts to score. 65

28 **Example 4 of play evolution - Outbound and Throw-in**: **(Left)** Ball was out-bounded, and Offender 1 has the ball. **(Middle)** Offender 1 is looking for the location to trow-in. **(Right)** Offender 1 throws the ball in. 65

29 **Example 5 of play evolution in Basketball - Turn over**: **(Left)** Yellow team just got the score at the right half-court (outside of the scene). **(Middle)** Most of the offenders and defenders are getting back to the left half-court. **(Right)** Offender 1 brought the ball into the left half-court (Turn over). 65

30 **Example 6 of play evolution in Basketball - Pass and Three point shot**: **Upper row: (Left)** Offender 1 just received the ball. a PoC appears at the location of the player, as the defender 3 is approaching to defend against the offender 1. **(Middle)** Offender 1 passes the ball to another offender 2 (Shooting Guard), and the defender 3 moves to the location of the offender 2 (another PoC appears at the offender 2). **(Right)** Offender 1 moves inside to receive the ball again (The first PoC moves to the location). **Lower row: (Left)** Offender 2 directly shoots rather than passes the ball to the offender 1. Defender 3 try to block the shot. **(Middle)** Because the ball is approaching to the basket, the new PoC appears in the region near under the basket. **(Right)** Finally the ball enters the basket and the PoC moves to the location under the basket. 66

31 **Example 7 of play evolution in Basketball - Fake and Drive-in shot**: **Upper row: (Left)**Offender 1 has the ball, and the defender 2 covers him. **(Middle)** Offender 1 dribbles past the defender. **(Right)** Defender 3 tries to cover offender 1. Other offenders at left and right side of offender 1 are ready to receive the ball. **Upper row: (Left)** Offender 1 pivots to drive by faking defender 3. **(Middle)** Offender 1 drives past defender 3, and PoC near basket appears. **(Right)** Finally the offender 1 successfully drives in. 67

32 **Example 8 of play evolution in Ice Hockey - Stickhandling and shoot**: **(Left)** Offender 1 (white team) stick-handles to the goalcage, and PoC appears near the player. **(Middle)** Offender 1 shoots, and the puck is about to pass by the goaltender. Additional PoC near the left side of the goal cage indicates the shot will not be successful. **(Right)** The puck is passed away and bounced to the upper side of rink. Some players are heading to the puck. 67

33 **Example of the quantitative evaluation for our approach**: **(Upper)** *Distances between the location of the detected* POC *in the current frame and the location of the ball in future frames.* The measurement varies from blue (4 frames later) through red (120 frames later). For both graphs, x-axis is the frame number, and y-axis is a pixel level distance (100 pixels is approximately 4 meters). **(Lower)** Ball Information, and game status (red : inbound, green : outbound) 68

34 **Pass and Interception:** The distance between current POC and the ball locations varying from current frame (Black plots) to 90 frames later (dark yellow plots). See the event depicted in Fig. 25 and Fig. 15 69

35 **Center Pass:** See the event in Fig. 26 70

36 **Through pass:** See the event in Fig. 27 70

37 **Out-bound and Throw-in:** See the event in Fig. 28 71

38 **Turn over:** See the event in Fig. 29 72

39 **Three point shot:** See the event in Fig. 30 72

40 **Fake and drive-in:** See the event in Fig. 31 73

41 **Stick Handling and Shot:** See the event in Fig. 32 74

42 **Gaussian Process Regression Flows (GPRF)**: **(a)** Video frames having trajectories, and the normalized *mean flows* representing 17 different *spatio-temporal patterns* of motion trajectories are visualized in different colors. Each mean flow is generated from trajectories extracted from videos using Gaussian process regression. We classify each online track by traversing inside the flow fields and collecting posterior densities along their paths. **(b)** Some of individual mean flows are shown : blue indicates that the mean flows in the region have lower variances (and thus higher certainty), and red indicates higher variances. 79

43 **Overview of our framework:** From tracks to GPRF. 79

44 **Example of GPRF, and mean flows**: **(Upper row) (a)** Sample trajectory having a duration of 16 frames: each arrow starts from the location of $x(u, v, t)$, and has the velocity values **(b)** The trajectory in (u, v, t) space $\in \Re^3$ **(c)** GPRF generated from the trajectory **(Bottom row)** The projected mean flows in each of $u - v$ slice at $t = 0$ **(d)**, $t = 8$ **(e)**, and $t = 16$ **(f)** respectively. Mean flows with different levels of confidence exist in any grid points in the space. In each image, only mean flows having the variance of less than half the maximum variance among 95% confidence are shown as color values. The colors vary from the larger posterior variances (red) to smaller variances (blue). 82

45 **Normalizing frames**: **(a)** Each red, blue, green and magenta line indicates the trajectories from tracks started from a same spatial position toward a same destination with different velocities, and accelerations (or in different frame rates). The image in the upper-right shows trajectories projected onto the spatial domain. **(b)** All of the trajectories are normalized in time (t) axis as l. **(c)** Another example showing that each colored line has the same trajectory with the same velocity and acceleration but diverges later. The dashed black line shows the normalized trajectory. 83

46 **Sampling points for training**: Suppose we are training a class using three trajectories. **(a)** All trajectories are normalized in l evenly distributed grid in time axis. **(b)** Adding all the samples per time grid for generating GPRF. In this case, training the same number of trajectories have to be guaranteed for all the other classes. **(c)** If we need only two samples per time grid, we randomly sample without replacement two points from three trajectories at each time slice. 84

47 **Measuring similarity with certainties**: A red dashed arrow indicates the approximation of learned trajectory. A blue dotted arrow is a mean flow vector at each test point and a black arrow indicates its velocity vector. In the right side of each image, posterior densities for each u, v, t are shown. **(a)** The test point x_a is close to ALT, but the vectors are quite different. **(b)** Both the location of the input point x_b, and its velocity are close to mean flow and ALT respectively. **(c)** The example of the worst case: The test point x_c is further away from ALT (large variances of the posterior probability densities), and the velocity vectors are different (away from mean). 86

48 **Kurtosis and Normality examples**: Graphs in the first and third rows show the distribution of sorted $\rho_k(\mathrm{T}_n)$ of some examples. Images in the second and the fourth row are the corresponding trajectories. In each graph, y–axis denotes the proportion of dominance (0.0 to 1.0), and x–axis shows grids of 17 different classes (shown in Fig. 42). The leftmost bar in each graph shows a dominance level of the chosen class. Listed with each graph are the excess kurtosis κ, global similarity $\mathcal{S}_k(\mathrm{T}_n)$, and normality value $\mathrm{N}_k(\mathrm{T}_\mathrm{n})$. Our method classified the examples in the upper row as normal and lower ones as anomalous trajectories. Notice that the third example in the upper row models a car which *stopped* for a while at the red light. 89

49 **Datasets and each of their GPRFs:** Example images from the videos used in this paper, and their learned GPRFs are shown. For each row, first column shows a normalized mean flows constructed from some of trajectories in each data set, and the second column shows testing trajectories, and the last column shows testing trajectories labeled in different colors for testing purpose. 91

50 **Trajectory recognition using incomplete trajectories:** For each graph, x–axis indicates the percentage of sub-tracks used for evaluation (i.e. 50% means by the first 50% of total length of testing trajectory). (**Upper row**) Results from our approach, and (**Bottom row**) Results from DDTW 93

51 **Similarity validation test:** In each image, the row indicates learned 17 classes, and the first 17 columns are the trajectories which are randomly chosen from each of labeled sets (for the visualization purpose, we arranged each trajectory corresponding to the order of classes). The last 6 columns consist of randomly chosen anomalous trajectories from our anomalous sets. Note that all values are normalized from 0 to 255 by the maximum similarity of each classes. (Left) Using global likelihood (Eq. 20), (Middle) Using Global Similarity (Eq. 22). (Right) Minimum distances from DDTW : a darker color represents a better score. 94

52 **ROC curves for anomaly detection:** x axis refers to False Positive Rate (FPR), and y axis for True Positive Rate (TPR), we evaluate each curve by varying a threshold on each normality function (Eq.25). **(Left) :** Adam, **(Middle) :** Ocean, **(Right) :** CLIF 95

53 **Qualitative evaluation of trajectory prediction:** Each arrow indicates a possible pattern of motion on the road. The color of each arrow indicates the probability level to be classified as the given pattern (See the color table in A.1). Dotted lines indicate the "stop (includes deceleration) and move" **A.1**. A car moves straight ahead with a probability of 67% for continuing to move straight. **A.2** As the car decelerates, a pattern for stopping situation becomes dominant. **A.3** The car slightly changes its direction to left. Because it needs to decelerate before a left turn, the probability of taking the left turn increases. **A.4** The left turn becomes dominant. **A.5** Finally, even before the car enters the left road, the probability for a left turn increases to 92%. **B.1** A car originally moves ahead in the second lane with the probability of 100% for continuing that motion pattern. **B.2** The car is slowing down. **B.3** Finally the car stopped for a traffic signal. **C.1** A car moves ahead to bottom left, but decreases its speed. Therefore even the spatial pattern is similar to straight, the patterns for parking becomes dominant **C.2** Finally the car parks. **D.1–2** Though spatial movement is similar to the examples shown in C, due to its velocity and acceleration, it was proven to be straight pattern. **E.1–2** and **F.1** Examples of prediction in UCF data 96

54 **Qualitative evaluation of incremental anomaly detection:** The color of the trajectory indicates the normalcy (green), and anomaly (red) **A.1** A police car goes in the wrong direction on a one-way road, and is classified as an anomalous pattern. **A.2** As the car merges to left direction, the trajectory becomes normal. **B** A car was suddenly stopped inside the intersection due to a pedestrian, and we classify it as an anomaly. **C** Bikes are moving inside the intersection. **D** A car takes a U-turn which is not allowed in the street. 96

55 **Overview**: **Top:** An example of the pan-tilt-zoom performed by a camera operator. The field of view changes from the region bounded by red lines to another bounded by blue lines, **Bottom (overhead view):** Arrows indicates a motion field generated only from the ground-motion of players with Gaussian process regression. The certainty level of the velocity vector at each location is represented by different colors. Circles indicate the predicted future locations of interest which can be interpreted as the location where the field of view of the camera will move. Color bar represents the level of values (Red denotes higher values). 98

56 **From input video to the detection of future important locations** 99

57 **Registration of each frame onto the reference view for a player tracking**: **Left:** Original view, **Right:** A rectified view used for tracking players 99

58 **Stochastic motion field and its certainty field generated from GPR:** The arrows indicate the vectors in the field generated from the motions of players. The colors of the arrows represent the level of certainty. Red arrows have larger certainty level (narrower confidence band) and blue ones have lower certainty level. Therefore extrapolated vectors are more likely to be blue. Bold white lines indicate the references for the ground plane. Note that all calculations were done in the overhead projection. 101

59 **Certainty transfer through stochastic motion field and merging points: Top-left:** The certainty level ρ at a location $\mathbf{x}_0$ is transferred to the location $\mathbf{x}_n$ through the stochastic motion field Φ. **Top-right:** In a separate grid Ψ, the value ρ is accumulated at the location of $\mathbf{x}_n$. Accumulated certainties in Ψ will be used to predict locations of future importance. **Bottom:** Colored circles indicate accumulated certainties from the motion field shown in Figure **??** (red circles with larger accumulations and blue ones with smaller accumulations). Note that we visualized the only locations that have more than 80% of maximum accumulation. 103

60 **Evaluation for the comparison of actual camera operator's field of view:** The region with solid yellow lines denotes the camera operator's field of view, whereas the region with dotted yellow lines represent the field view 10 frames later. The convex hull of only the locations of players is shown with purple lines. The region with red lines is decided by the locations of players and the merging points computed by GPR. 104

61 **Quantitative evaluation for the comparison of actual camera operator's field of view:** The values in vertical (y)- axis in both graphs indicates a Jaccard coefficient between a camera operator's region and each computed region, which uses the location of players (black), the 2D RBF (red), the 2D GPR (green), and the 3D GPR (violet) respectively. A graph in the top shows the evaluation over all frames from one sample from our data set (cv5). The bottom graph shows the average of Jaccard coefficient for all the football data sets (cv1 to cv8). 106

62 **Evaluation of the future region of camera operator by differing frames:** We evaluated each data set (cv1 to cv6 are shown here) by comparing the field of view of camera operator in future frames with the predicted merging regions at current frame by differing the frame difference from +10 to +40 frames. As shown in each figure, the method using only tracked player locations has lower similarity, and is generally decreasing as it always stay at the current location of players while methods using motion field have larger similarity over different frame offsets. In the results from cv4 and cv6, both GPR and RBF methods give similar results, while the other data sets show that GPR-based approaches work better. In the two data sets (cv4 and cv6), because the actual region of interests are close to boundary and fewer extrapolated vectors are involved in the prediction, both GPR and RBF methods give similar results. 107

63 **Computational expense:** Red line indicates the computational expense of 2D RBF-based approach. Green indicates those of 2D GPR-based approach, and violet line indicates those from 3D GPR-based approach. x axis refers the each data set, and y axis refers the millisecond. 108

64 **Qualitative evaluation between RBF method and GPR method: Top row** shows the transition (PTZ) of the original views adjusted by the camera operator. To give a better understanding of how the original view moves, we added white lines to represent the 50 yard line and the upper boundary of the ground field. The view is being panned to the right direction, and zoomed out. **Middle row** shows the registered over-head projection of the stochastic motion field, and merging points computed from the 2D GPR method. **Bottom row** represents the result from the RBF based approach. Note that the merging points in RBF method are often concentrated near the boundary of the field because the computation of the merging points are highly affected by the extrapolated vectors in RBF (see the last example of the third row). 111

65 **Additional comparison of our results (with GPR method) and actual camera motion:** Sequences in **Top row** demonstrate how our approach mimics pan and zoom-out. For the sequence at **Bottom row**, as merging points staying at the center of field of view, then move away from the camera operator, the field of view changes from panning to zoom-in. 111

66 **Qualitative comparison between RBF method and 2D GPR method:** The sequence of scenes in **top row** show the result from RBF method (Chapter 6); the red contour indicates the location where the motion field merges. The scenes in **bottom row** show the result of our approach using GPR. The location of where the motion field merges is shown with circles in which the colors represents the amount of accumulated transferred certainties. For each row, first scene describes the merging location lies in front of player **A** who dribbles the ball. In the second scene, merging location describes the location where the other offender **B** will receive the ball. In the last scene, results shows the location for the other pass. 112

CHAPTER I

ABSTRACT

Analysis and visualization of dynamic scenes is often constrained by the amount of spatio-temporal information available from the environment. In most scenarios, we have to account for incomplete information and sparse motion data, requiring us to employ interpolation and approximation methods to fill for the missing information. Scattered data interpolation and approximation techniques have been widely used for solving the problem of completing surfaces and images with incomplete input data. We introduce approaches for such data interpolation and approximation from limited sensors, into the domain of analyzing and visualizing dynamic scenes. Data from dynamic scenes is subject to constraints due to the spatial layout of the scene and/or the configurations of video cameras in use. Such constraints include: (1) sparsely available cameras observing the scene, (2) limited field of view provided by the cameras in use, (3) incomplete motion at a specific moment, and (4) varying frame rates due to different exposures and resolutions.

In this thesis, we establish these forms of incompleteness in the scene, as spatio-temporal uncertainties, and propose solutions for resolving the uncertainties by applying scattered data approximation into a spatio-temporal domain.

The main contributions of this research are as follows: First, we provide an efficient framework to visualize large-scale dynamic scenes from distributed static videos. Second, we adopt Radial Basis Function (RBF) interpolation to the spatio-temporal domain to generate global motion tendency. The tendency, represented by a dense flow field, is used to optimally pan and tilt a video camera. Third, we propose a

method to represent motion trajectories using stochastic vector fields. Gaussian Process Regression (GPR) is used to generate a dense vector field and the certainty of each vector in the field. The generated stochastic fields are used for recognizing motion patterns under varying frame-rate and incompleteness of the input videos. Fourth, we also show that the stochastic representation of vector field can also be used for modeling global tendency to detect the region of interests in dynamic scenes with camera motion. We evaluate and demonstrate our approaches in several applications for visualizing virtual cities, automating sports broadcasting, and recognizing traffic patterns in surveillance videos.

CHAPTER II

INTRODUCTION

Dynamic scene analysis has become one of the most important and active research areas upon the widespread availability of public and personal capturing devices. Extracted dynamic information from such devices can be used for a variety of applications, including intelligent surveillance, crowd-casted multimedia, and augmented reality systems. Therefore, the role of research in managing/analyzing dynamic information from the collection of various types of videos is increasing in importance because of its direct impact on many real-world applications.

In general, the field of view (FOV) of a conventional camera can capture only a limited amount of visual information. Therefore, analyzing and visualizing the motion of moving objects requires meeting one of the following conditions: (1) When a camera can pan and tilt, it should be able to track objects and adjust its own view to maintain the objects within its FOV; (2) when a camera is static, other cameras have to cover the possible regions into which the objects may move; and (3) when a camera is at a distance, it has to cover a larger region with a higher resolution.

Unfortunately, it is quite difficult to set up such conditions in real-world situations for several reasons. First, if a large number of moving objects occupy a scene, selecting adequate targets and regions of interest is necessary; however, the local tracks of individual objects may not always identify the right objects or regions of interest, calling for an additional algorithm for adequate region selection. In addition, tracking results (which indicate the current location of objects) do not guarantee the stable localization of an FOV in case the target objects move rapidly. Second, for static cameras, covering possible regions where target objects could appear may require

a large number of video cameras, which is difficult, if not impossible to set up in practice. Finally, cameras at a distance usually tend not to provide a higher resolution of moving objects, which is necessary for the visual details of objects. Although ultra definite video streams from aerial or satellite cameras provide a wider view with a higher resolution, they could still suffer from a low-frame rate, which is cumbersome for temporal analysis. Against the backdrop, the analysis of dynamic scenes under conditions of sparse data over space and time raises crucial research questions.

To tackle the research questions and provide solutions needed for the analysis, we applied scattered data interpolation and an approximation method to a spatio-temporal domain. During the last decade, scattered data approximation (SDA) and scattered data interpolation (SDI) have widely been used for spatial data inference [82, 8]. These approaches were used for various applications including surface reconstruction from sparse vertex data [20, 21, 131], geostatistical analysis from sparsely sampled measurements [56, 46, 124], and image completion and inpainting [58, 23]. While the applications interpolate and approximate unobserved data from spatially sparse observation, in this thesis, we applied similar approaches in the space and time domains to predict motion information in the input videos. The following summarizes the examples of the analysis that we introduce in this thesis:

Dynamic Scene Visualization from Sparsely Distributed Videos Recently, public/private video cameras such as those for surveillance and the monitoring of traffic, are ubiquitous in cities. Most of them, however, are generally static and sparsely located. For an analysis of video data from such sparsely-distributed static cameras, we apply various data interpolation techniques for the motion estimation of objects in unobserved regions for completing the dynamic visuals of large-scale scenery [69, 70]. Based on the conditions and prior knowledge of the structure of the target scenes

and types of motions of objects, we generate a prototypic system for city-level visualization using a collection of videos (traffic, surveillance, and personal videos). This research is presented in Chapter 5.

Global Motion Analysis for Adjusting the View points of a Camera It is not easy to visualize dynamic scenes with a limited number of cameras in the case of many moving objects with complex movements, whose group behavior cannot easily be modeled.

Therefore, we propose a method of providing an effective way of controlling the cameras in order to adjust their FOV. The main goal of this research is to create an automated broadcasting system in which an intelligent system mimics an actual camera operator that is capturing dynamic sports scenes. The approach mainly focuses on discovering important locations, given the global behaviors of players on the field. We first calculate the motion fields, representing the global movement of the players from *spatio-temporal radial basis function interpolation (RBF)*. We then calculate the location of importance in the near future by detecting where the global motion flow merges [67, 66]. The application and the details of extracting stable motion and constructing a motion field using RBF are introduced in Chapter 6. In addition, calculating the location of importance using stochastic regression is also proposed in Chapter 8 with more extensive quantitative evaluations.

Motion Pattern Recognition and Anomaly Detection from Spatio-Temporally Sparse Scenes As discussed earlier, motion data in a practical situation for a surveillance system can be sparse both spatially and temporally. For example, a realtime surveillance system may have to determine a pattern of a given motion trajectory or its normalcy even with incomplete trajectories at a specific moment. In addition, the frame rate of the input video could vary depending on the type of camera (i.e., aerial

or satellite videos) or the various exposures (i.e., day and night). Therefore, an effective prediction of patterns of an incomplete motion, regardless of its frame rate, is one of the biggest challenges in the domain of intelligent surveillance systems. It is here that the method we propose to predict patterns and detect anomalous motion fits in, even with spatio-temporally sparse input trajectories using *Gaussian process regression flow (GPRF)* [68]. This approach successfully infers the traffic patterns from a variety of data sets collected from different types of frame rates and conditions. The approaches and results are presented in Chapter 7.

2.1 Thesis Statement

"Scattered data approximation techniques in the spatio-temporal domain provide effective solutions for dynamic scene analysis and visualization."

In particular, we propose to apply scattered data approximation techniques for predicting the motions and the behaviors of moving objects. We provide the following contributions for supporting the thesis statement:

1. A method to analyze and visualize dynamic scenes in a city with sparsely distributed cameras [69, 70].

2. An algorithm to detect the regions of interest from sparse sets of motions captured from both static and moving cameras [67].

3. A novel representation of motion trajectories with stochastic motion field [68].

4. A technique to classify motion trajectories with incomplete motions under sparse frame-rate [68].

2.2 Outline of the Remaining Chapters

The remainder of this thesis proceeds as follows: Section 3.1 briefly summarizes the background and related work on dynamic scene visualization and analysis. Section 4

introduces scattered data interpolation and approximation methods and related studies using such methods in the spatio-temporal domain and provides several examples for dynamic scene analysis, regarding real-world situation described in earlier section 2. Through Chapter 5 we show how data interpolation is used for large-scale scene visualization using distributed static cameras. Chapters 6 describe how we predict the location of important region in a complex dynamic scene by constructing a dense global motion field using RBF. Chapter 7 introduces a method that represents motion trajectories in surveillance footage as a stochastic vector field, which resolves the sparse and incomplete problem in the real-world situation. Then in Chapter 8, we introduce a method to detect regions of interest in dynamic sports scenes captured from moving cameras. In the chapter, GPR introduced in Chapter 7 is used for motion field analysis under the same objective which is introduced in Chapter 6. We also evaluate our method by the comparison between the region detected by our method and the region decided by actual camera operators.

CHAPTER III

RELATED WORK

In this chapter, we summarize the related work and previous efforts in the areas of dynamic scene analysis and visualization. We first introduce generic vision and graphics techniques needed for video analysis. Then we describe the comparison between various types of scene visualization techniques. Finally, we briefly introduce scattered data interpolation/approximation methods which construct the essential framework for solving sparse data problem in the dynamic scene analysis and visualization.

3.1 Methods for Dynamic Scene Analysis

To analyze the dynamic scenes in input videos, we must understand the underlying context of the components inside the video frames (e.g., moving objects and background scenery). One of the key preprocessing tasks for such an analysis is to locate the moving objects in the dynamic scene. We first extract the *segments of the foreground*, which represent the screen-space region of the moving objects, and then *track* their spatial locations, appearance, poses, or surfaces. Knowing the motions and behaviors of a given object is also useful for the analysis of dynamic objects. Visualization of background scenery often requires the *structure of a stationary background* of the scene or even rough geometric information regarding the cameras in use. In the following subsections, we briefly introduce the related methods and algorithms.

3.1.1 Foreground segmentation

For the case of a stationary camera, the background of a scene usually contains stationary structures while the moving objects change their locations and poses. In such a case, since the background undergoes fewer changes in the scene over time,

an Eigenbackground model [97], a mixture of Gaussian model [125, 83], and their variations [77] are generally used for video taken by a fixed camera.

However, if the pose and the physical location of the camera changes over time, the assumption of a stationary background no longer holds and the approaches of a fixed camera do not work properly. Therefore, in such a case, a variety of patch-based image segmentation methods are used. Such approaches typically use (1) the similarity among low-level features (bottom up) [29, 119] or prior knowledge or context (top down) [79, 22]. However, applying such algorithms frame by frame to videos incurs high computational expense. Recently, the study of video segmentation for moving cameras has undergone several advances [141, 48, 63] resulting from the exploration of the relations between spatial segmentation and temporal consistency.

3.1.2 Tracking Objects in a Scene

To consistently recognize the locations of moving objects in the scene, one has to apply a tracking algorithm to the video. Although several methods of tracking moving objects are used, methods vary depending on the purpose of tracking, the properties of the target objects, and the camera settings.

One of the most well-known feature-level optic flow-based trackers is the KLT Tracker [17]. This tracker first detects the low-level features with higher saliency [120] and then tracks correspondences deterministically [135, 17]. Similar yet various feature detectors and trackers are also used for plane tracking and structure from motion which will be introduced in Section 3.1.3.

If the initial region (or patch) to track is assigned (i.e. segmentation) and the region has particular properties such as color, feature space and gradients, one can also use Kalman filters [59, 57] or particle filters [55] to track a consistent trace of moving objects using probabilistic state-space models without any deterministic settings. These approaches are more robust even for occlusion handling and multiple

targets by multi-hypothesis frameworks typically based on particle filtering [65].

Multi-view tracking can also be performed by several methods [71, 39, 67] that are more robust in occlusion handling and have fewer geometric errors when the tracking results are evaluated in a ground coordinate [39, 67].

3.1.3 Estimating Scene Geometry

In many scene visualization applications, registration is a key method for constructing scenery or events extracted from videos. Therefore, extracting projective geometries in a scene by estimating planes and 3D structures is an essential task.

To handle the projective geometry of planes, finding homographies [50, 1] that project the same plane from different views is a basic task for plane registration. Another key element for estimating perspective geometry without full camera calibration is to estimate vanishing points in perspective views [73, 114]. If dense views with sufficient motions of cameras or a pair of stereo views are available, a three-dimensional reconstruction of a static scene can be done by a variety of structure from motion algorithms [87, 60, 117]. The reconstruction of scene geometry has been demonstrated in many applications [122, 2].

For scenes with complex structures with insufficient views/motions, user interaction tools have also been used for scene reconstruction [53, 33].

3.1.4 Motion and Flow Field Analysis

A flow field, usually a two- or three-dimensional vector field generated from motion vectors, is often used for examining the motions of objects or moving planes [136, 67]. Methods of generating a flow field from optic and gradient domains have been introduced in many motion analysis studies [136, 107]. Once data have continuous vector fields, then the field can be decomposed into several parts, which reveals the unique behavior of flows that appear to be irregular [129]. In addition, calculating critical (singular) points in the flow field sometimes unveils its characteristics and

even defines the types of flow fields and their identification [30, 129, 28].

3.1.5 Behavioral Models

A large body of research in the field of robotics has analyzed individual or group behavior [11, 7]. The field of graphics also applies behavioral analysis in character animation and crowd simulation [111, 112, 130, 44, 42]. These behavioral models usually adopt the characteristics of an individual agent with behavioral dynamics [42], knowledge-based and cognitive learning [44], a predefined rule-based approach [111, 112], and sometimes a potential field [130] in which group behavior can be simulated or predicted. Chapter 6 shows the analysis of group behavior can be used for effective dynamic scene visualization.

3.2 Scene Visualization techniques

A technique required for creating a better view of the given scene is scene visualization, which provides viewers with information about the scene. The key challenges for scene visualization are to define the target object (or geometry), and how to create better visuals within the given input resources.

In the last decade, researchers in computer vision and graphics have proposed a number of techniques for scene visualization. They consist of reconstructing 3D objects [80, 47, 18, 92, 32, 138, 75, 76] or three-dimensional scene geometry [122, 2, 123], synthesizing views [85, 40, 134, 118], and extending the duration of video resources over time and space [115, 3, 93]. The output of each method varies depending on the purpose and target objects of the visualization, and the given settings of the resources.

Table 1 describes the differences among the various methods used for scene visualization. However, as shown in the table 1, the approaches that handle less dynamic information **(DI)** property are not adequate for dynamic scene visualization from a practical standpoint. They need for a larger number of cameras (**CS :camera**

Table 1: Categorization of methods used for scene visualization **The top** table represents terms that characterize each property that must be account for by scene visualization. Each property can be ascribed a low or high values. **The bottom** table enumerates various methods for scene visualization, and locates our methods at the bottom of the table. †: The rigidity of target objects, ‡: This indicates both the temporal continuity of a given scene and the dynamism of the target object

Alias and description of properties for methods used for Scene visualization				
Alias	**Description**	**(L)**ow	**(M)**edium	**(H)**igh
CS	Camera Sparsity	Single View	Sparse Multi-view	Dense Multiview
SC	Spatial Scale	Indoor (lab)	Outdoor, limited FOV	Outdoor, City level
RG	Rigidity †	Single Rigid	Complex rigid	Deformable
DI	Dynamics ‡	Stationary	Limited, Repetitive	Fully dynamic
RS	Resolution	Small	Medium	High or adaptive
VA	Viewing Angle	fixed	limited with offsets	Full DOF

Related and Previous works							
Methods	**Reference**	**CS**	**SC**	**RG**	**DI**	**RS**	**VA**
Light Field & Lumigraph	[80, 47]	H	L	M	L	L	M
Unstructured Lumigraph	[18, 92]	M	M	L	L	M	M
VBR, Billboard, Volumetric	[13]	H	M	H	H	M	M
LiberoVision and Eye Vision	[85, 40]	M	M	H	L	M	M
Video Texture	[115],	L	L	H	M	L	L
Panoramic Video Texture	[3],[93]	H	M	M	M	H	M
SfM, 3D reconstruction	[122, 2, 123]	H	H	M	L	M	H
Visual hull, Silhouette	[32, 138, 75, 76]	M	L	H	H	M	H
View Morph, interpolation	[134, 118]	M	L	L	L	L	M

sparsity) also restricts the flexibility of the application because of the problem of sparsity, which has already been discussed in an earlier chapter. The lower resolution of outputs (**RS : resolution**) and viewing angle flexibility (**VA : viewing angle**) also bring the limited quality of the visualization. Therefore, this thesis will describe the scene visualization approaches that apply more DI, less CS, a larger VA and RS, and various SC (spatial scale) based on dynamic scene analysis, particularly for the visualization of a city, discussed in Chapter 5.

3.3 Interpolation and Scattered Data Approximation

Data interpolation is a method of constructing new data points and determining their values within the range of a discrete set of known data points. The known data points are often obtained by samples or observations from experiment. Depending on the technique needed for the determination of the value and the properties of the data points, the interpolation method takes on a variety of forms, including constant [12], linear (or bilinear) [89], polynomial [16], and spline interpolation [139].

Scattered data interpolation (SDI) and approximation (SDA) [82] are the methods used for cases in which the data locations and the influence from each point are unstructured and fro cases in which we do not have any information about how regular the grid is and how dense the data are. The scattered data interpolation for an unorganized data process has commonly been used for solving many image processing and graphics problems such as surface reconstruction [21, 131] and image restoration [100, 150, 58]. It is also widely used in the field of geo-statistical analysis [46, 124]. The most popular methods have been variations of radial basis function (RBF) interpolation [21, 131, 19]. Recently, approximation using a stochastic process such as Gaussian process regression (GPR) (also known as "kriging") has become popular in various areas [108, 8, 46, 56, 52] for its compactness and powerful

inference mechanism for missing data. In Chapter 4, we first describe the mathematical form of the two most popular SDI/SDA methods and then briefly introduce the relationship between RBF and GPR. In Section 4.3, we show that applying the SDI/SDA techniques to the spatio-temporal domain is an effective way for dynamic scene analysis.

Then in Chapters 5, 6, 8, and 7, we present applications and details of how we utilize such spatio-temporal scattered data approximation for video data with different data-sets and answer various research questions.

CHAPTER IV

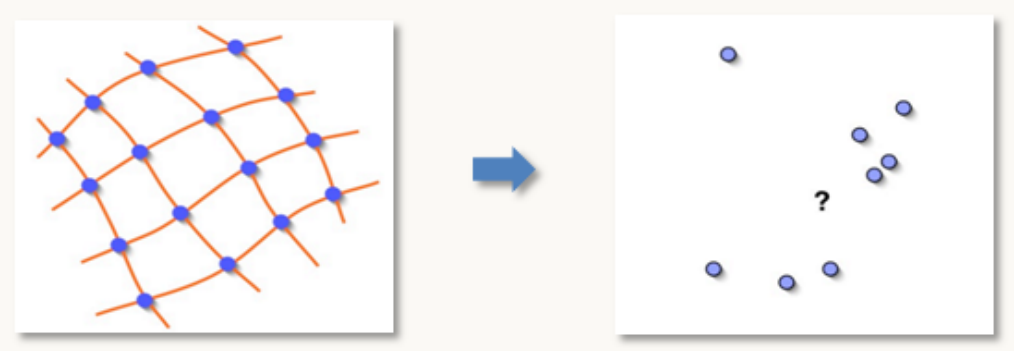
SCATTERED DATA INTERPOLATION AND APPROXIMATION

As briefly introduced in the Section 3.3, a combination of scattered data interpolation and approximation (SDI/SDA) is a method of inferring values from the observed (or trained) data points that fall on an unorganized grid.

Mathematically, the basic formulation of the scattered data interpolation problem can be shown as follows. Let f be an unknown function that maps $\Re^d$ to $\Re^1$. Now suppose that $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is a set of points $\in \Re^d$, in which the d is a dimension of the input points. Now if $f_1, \ldots, f_n$ is a set of values $\in \Re^1$, which is a set of corresponding values for $\mathbf{x}_1, \ldots, \mathbf{x}_n$, then the goal of SDI is to find a representative function $\hat{f}$, such that $\hat{f}(\mathbf{x}_i) = f_i$ $(1 \leq i \leq n)$. For the case of scattered data approximation, we often consider the residual ϵ, such that $\hat{f}(\mathbf{x}_i) = f_i + \epsilon$ $(1 \leq i \leq n)$. We can deal with this residual with a form of regularization in the regression problem [78] or by considering input and output data as a stochastic process [108].

However, the above basic SDI formulation could be satisfied by a number of functions. The representative function could even be a constant function $\hat{f}(\mathbf{x}) = f_i$ if $\mathbf{x} = \mathbf{x}_i$ and 0 otherwise. Therefore, we generally assume some properties of function $\hat{f}$. First, it would follow some smooth properties such as continuity or differentiability. Secondly, it could belong to a family or class of function sets. Finally, we could also construct implicit conditions such as a boundary condition. Therefore, we assume that f can be well represented within a function space $\mathcal{F}$ [82], which allows us redefine the SDI problem as follows: Find a function $\hat{f} \in \mathcal{F}$, such that $\hat{f}(\mathbf{x}_i) = f_i$ $(1 \leq i \leq n)$. Figure. 1 illustrates a simple SDI/SDA regarding a geometric property.

This formulation in the function space is a general step for many interpolation and approximation problems, based on the properties for describing function $\hat{f}$ shown above. In addition, many types of SDA/SDI methods have already been discussed in a previous chapter [89, 16, 139, 82]. Among the various types of SDI/SDA, RBF and GPR method is introduced and formulated in the following sections.



(a) (b)

Figure 1: Example of a simple scattered data interpolation: (a) observed scattered data points (shown in blue) regarding geometric properties (e.g., the surface) and (b) a prediction of the values at unknown locations.

4.1 Radial Basis Function (RBF) Interpolation

The most commonly used scattered data interpolation method is the radial basis function (RBF) which consists of a finite linear combination of translated basis functions (kernel functions) that are a radially-symmetric function of the form $\phi(\|\cdot\|)$, where $\|\cdot\|$ is the Euclidean norm. Now the general form of the interpolation function is shown as follows:

$$\hat{f}(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{n}} \lambda_{\mathbf{i}} \phi(\|\mathbf{x} - \mathbf{x_i}\|), \tag{1}$$

where $\lambda_i \in \Re$ is the coefficient (weight) of the basis function, and $\mathbf{x}_i \in \Re^d$ a known data point. Intuitively, we can think of each weight and basis function of the norm between each scattering of data points as the influence between the data points and their values for the interpolation.

For the purpose of the interpolation, we choose the basis functions to emphasize desirable qualities such as continuity or differentiability. For example, the thin-plate spline RBF, derived from the bending energy function that represents the integration of the second-order derivatives by Duchon [34], is a good choice considering its geometric properties (e.g., C^1 continuity). The following lists commonly used basis functions:

i Biharmonic basis: $\phi(r) = r$

ii Thin-plate spline basis : $\phi(r) = r^2 \log r$

iii multi-quadrics basis: $\phi(r) = \sqrt{r^2 + c^2}$

iv Gaussian basis: $\phi(r) = \exp(-(r/c)^2)$

Several of the above basis functions have dimensional constraints. For example, thin-plate and Gaussian basis functions provide only smooth interpolation in a dimension of less than 3 [34]. Once the basis functions are chosen, the coefficients can be calculated by solving linear system $\mathbf{A}\mathbf{\Lambda} = \mathbf{f}$, in which matrix $\mathbf{A}$ is a square matrix with each entry as $\phi_{j,k} = \phi(\|\mathbf{x}_j - \mathbf{x}_k\|)$, vector Λ has weight λ_i, and vector $\mathbf{f}$ represents the values of scattered input data points.

Sometimes, a more accurate approximation can be obtained by adding a polynomial term into Eq. 1. Adding a term is more effective for several reasons. For one, approximation often *reproduces* polynomials up to a given degree. For example, in a specific case in which a deformation at the sample point follows affine, we may need an RBF approximation that yields an affine function with a polynomial of degree 1. In addition, we sometimes want to decrease an *extrapolated* value to zero, which is far from the data points. In this case, an additional polynomial value is used for modeling the far-field behavior of the interpolated value. Finally, the polynomial is also useful when the RBF kernels handle the null space. Therefore, a radial basis

function interpolation/approximation with a polynomial term can be rewritten as:

$$\hat{f}(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{n}} \lambda_{\mathbf{i}} \phi(\|\mathbf{x} - \mathbf{x_i}\|) + \mathbf{g}(\mathbf{x}), \tag{2}$$

where g is a set of real-valued polynomials of p variables and with degrees of at most $m-1$. Now, function $\hat{f}$ is not only determined by weights λ_i but also the coefficients of the polynomials in g whose function still depends linearly on both coefficients. In this thesis, most of the work utilizes Eq. 2 as a general formulation of RBF interpolation and approximation. The details of the calculation of RBF interpolation needed for each task will be described in each application chapter.

4.2 Gaussian Process Regression (GPR) (Kriging)

As already discussed in the beginning of this chapter, when we predict or interpolate unknown values from the set of observed data (scattered data points), we must perform a number of activities such as choosing the right kernel, finding a method that deals with regularization regarding geometric properties. While a good choice among the variation of radial basis functions yields a reasonable results, it depends on the specific characteristics of the input observation (i.e., data sparsity and increasing/decreasing properties) or even for the parametric/non-parametric characteristics of the chosen basis functions.

Stochastic processes allow room for modeling input observations and output predictions as random variables. Among them, the Gaussian process has recently been applied in many fields because it provides not only interpolated/approximated values but also a probability of the certainty of the interpolation [110]. In addition, it provides an effective mechanism that optimizes hyper-parameters in a kernel (or a covariance function) into a form of a likelihood measure.

In this section, we introduce and formulate a method of approximating values from scattered observations using Gaussian process regression (GPR), and in Section 4.2.2,

we will also describe the relationship between generic RBF methods and the GPR method.

4.2.1 Gaussian Process and Data Prediction

A Gaussian process is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution [108]. Thus, it is completely specified by its mean and covariance functions. We first define a mean function $m(x)$ and a covariance function $K(x, x'')$ of a real process $f(x)$ as:

$$m(x) = \mathbb{E}[f(x)], \tag{3}$$

$$K(x, x'') = \mathbb{E}[(f(x) - m(x))(f(x'') - m(x''))] = \mathbb{E}[f(x)f(x'')] \tag{4}$$

Because function values are typically not directly accessible when we infer unknown values for the approximation problems, we consider a regression model with noise term ε : $y = f(x) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ and $f(x)$ is a zero-mean Gaussian process with covariance function $K(x, x'') : \Re^d \times \Re^d \to \Re$. Then, the observation process $y(x)$ is a zero-mean Gaussian process with covariance function $K(x, x'') + \sigma^2\delta_{p,q}$, where $\delta_{p,q}$ is a Kronecker Delta which is one iff $p = q$ and zero otherwise.

If we have training data $D = \{x_i, y_i\}_{i=1}^N$, the $N \times N$ covariance matrix $\mathbf{K}$ is now defined as $[\mathbf{K}]_{jk} = K(x_j, x_k)$. We then define the observation vector $\mathbf{y} = [y_1, \ldots, y_N]^T$; $\mathbf{y}$ can be shown as a zero mean multivariate Gaussian process with a covariance matrix:

$$\mathbf{K}^* = \mathbf{K} + \sigma^2\mathbf{I} \tag{5}$$

Thus, regarding the prediction of a value on the unobserved point x^* in the regression model f, our task is to compute posterior density $p(f^*|x^*, D)$. We then write the joint distribution of the observed target values in $\mathbf{y}$ and the function values $\mathbf{f}^*$ at the testing location:

$$\begin{bmatrix} \mathbf{y} \\ \hline \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} + \sigma^2\mathbf{I} & \mathbf{k}(x^*)^T \\ \mathbf{k}(x^*) & K(x^*, x^*) \end{bmatrix}\right) \tag{6}$$

,where $\mathbf{k}(x^*) = [K(x^*, x_1), \ldots, K(x^*, x_n)]^T$.

Based on the joint density, we can derive the posterior density $p(f^*|x^*, D)$ [108], shown as:

$$p(f^*|x^*, D) \sim \mathcal{N}(\mathbf{k}(x^*)^T(\mathbf{K}^*)^{-1}y \ , \quad K(x^*, x^*) - \mathbf{k}(x^*)^T(\mathbf{K}^*)^{-1}\mathbf{k}(x^*)) \tag{7}$$

Now we can summarize the mean and variance of the prediction on testing point x^* for the estimated function value f^* as:

$$\mathbb{E}[f(x)] = \mathbf{k}(x^*)^T(\mathbf{K}^*)^{-1}y \tag{8}$$

$$\mathbb{V}\text{ar}[f(x)] = K(x^*, x^*) - \mathbf{k}(x^*)^T(\mathbf{K}^*)^{-1}\mathbf{k}(x^*) \tag{9}$$

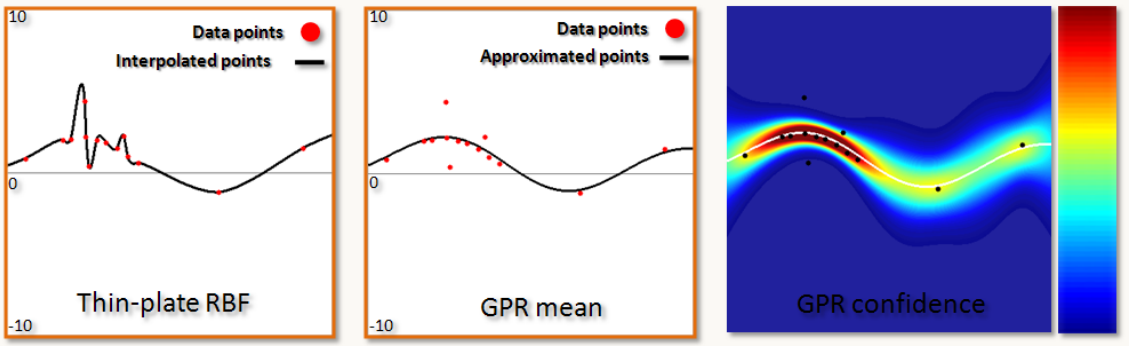
4.2.2 Gaussian Process Regression and Radial Basis Function

Given Eqs. 8 and 9, the predictive distribution from Gaussian process regression can be interpreted as a form of a weighted sum of a kernel. Note that the mean prediction in Eq. 8 is a linear combination of observations $\mathbf{y}$, often referred to as a *linear predictor*. In addition, if we think of the covariance matrix with a noise term, shown as $\mathbf{K}^*$, a kernel function, the equation can be interpreted as a linear combination of n kernel functions. Therefore, the mean of prediction $\mathbb{E}[f(x)]$ with testing point x^* can be represented as:

$$\mathbb{E}[f(x)] = \sum_{i=1}^{n} \lambda_i k(x_i, x^*) \tag{10}$$

As described in Section 4.1 RBF has a well-structured interpolation mechanism regarding geometric properties with a regularized form. However, while both have similar mathematical forms (i.e., a linear combination of kernel functions), many choices of covariance are not radially symmetric, and the choices of a radial basis cannot be covariances. Recently, Anjo [8] showed that these two approaches could be a mathematically common interpolation framework in a function space analysis (i.e., reproducing kernel Hilbert spaces (RKHS)). However, as GPR yields a probability of the certainty (as a form of variance; Eq 9) of the approximation, and provides a more

flexible choice for choosing a kernel as a form of covariance at any dimension [82], GPR may be a good choice for cases in which inputs with very noisy observations are used, and for cases in which the prediction is critical for other purposes (i.e., recognition from interpolants or outlining extrapolation without extra constraints). By contrast, RBF may be a good choice in specific cases in which the input observation is stable and a geometric property of the interpolation is known beforehand, as long as the right basis was chosen. We will discuss these issues in separate chapters with different objectives. Figure. 2 illustrates the interpolation and approximation of a simple 1D-example using thin-plate RBF and Gaussian process regression using a squared exponential covariance function.



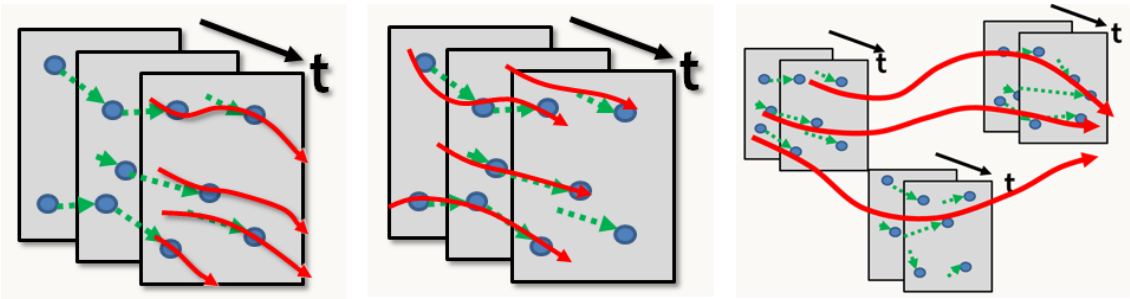
(a) (b) (c)

Figure 2: One-dimensional example of a scattered data interpolation using thin-plate RBF and GPR: The red points in (a) and (b) denote input data points (x axis) and their values (y axis), the black curves indicate the interpolated/approximated values (a) interpolated values using thin-plate RBF; keeping C^1 continuity as a geometric constraint (b) approximated values (mean) from GPR, and (c) a color-bar in the right side represents a certainty level (values indicate the band size in a Gaussian distribution of 95 percent based on its variance); red indicates a higher certainty (small variance); and blue indicates a lower certainty (large variance)

4.3 Spatio-Temporal Scattered Data Interpolation and Approximation

Sections 4.1 to 4.2 briefly introduced the RBF and GPR methods and described how they were related and how they were used for scattered data interpolation and approximation. In this section, we show how the methods can be used for dynamic scenes and what kinds of research questions that we can answer using these methods in a variety of dynamic scene analyses.

Figure. 3 shows the abstracts of various applications of SDI/SDA in dynamic scene analysis. As discussed earlier, we are interested in deriving a denser representation of sparse motion information in various ways (e.g., sparse over cameras or over frames). Thus, our preliminary task is to generate a dense vector field from motion information for each frame (Figure. 3 (a)) or from spatio-temporal domain (Fig. 3 (b)) or even from different views (Fig. 3 (c)).



(a) (b) (c)

Figure 3: Abstracts of spatio-temporal SDI/SDA in a dynamic scene domain Examples describe how SDI/SDA is applied to dynamic scene analysis. For each image, each rectangle indicates a frame in a video. Blue dots indicate all types of features, including detected/tracked objects in video frames. Green dotted arrows show a velocity vector based on the corresponding features between frames. Red arrows represent interpolated motions: (a) the generation of a continuous dense vector field in a frame at time t using observations from sparse motion data collected from previous frames, (b) a spatio-temporal dense field describing motion tendencies over a specific duration, and (c) propagated dense motion information even in unobserved locations

From the generated dense vector fields, our first application deals with data interpolation/approximation as a propagation of information from other observations. This type of approach will be introduced in Chapter 5. In the second application, we identify global motion tendencies from a set of sparse motions in the video. Because the motions from multiple moving objects are variable and complex, an storing understanding of the global motion tendency from sparse motions can facilitate the decision about where the observation (field of view) has to be changed for a better representation of the dynamic scene. Further details about this approach will be introduced in Chapters 6 and 8. Finally, data interpolation and approximation of temporally sparse motions can be effective at recognizing motion patterns in various video conditions, which include videos with a varying frame-rate and those with incomplete motion data. This approach will also be introduced in Chapter 7. In addition, as we already discussed in Section 4.2, the extracted confidence level from the variance of GPR provides information about how certain the approximated values are from the sparse number of observed data. The benefits from the use of this certainty information for dynamic scene analysis will also be shown in Chapter 7 and 8.

CHAPTER V

DYNAMIC SCENE VISUALIZATION FROM SPARSELY DISTRIBUTED VIDEOS

In this chapter, we introduce methods for augmenting aerial visualizations of Earth (from tools such as Google Earth or Microsoft Virtual Earth) with dynamic information obtained from multiple videos. However, because the video cameras are often scattered and each camera has a limited field of view, we have to consider scattered data problems (already discussed in previous chapters) from the distributed cameras for the analysis of the objects in each video. In addition, the configuration of such distributed cameras can differ from one scenario to the next. For example, we use sparse videos of clouds taken from various locations for the visualization of moving clouds in AEM. We may want to visualize pedestrians shown in a video taken from a single-view surveillance camera and show traffic motions from sparsely-distributed traffic cameras. Furthermore, in each of the above instances, we have to deal with different viewpoints of the videos, and in many cases, with incomplete information.

To provide adequate solutions for the scattered data problems, we divide our methods into four scenarios that address the distribution of cameras and the motions of objects. The scenarios are discussed in detail below.

#1 Direct Mapping (DM): A video is analyzed directly from a single view surveillance camera. Extracted data from the video are projected onto limited regions covered by the camera's field of view. We showcase several examples of people walking around.

#2 Overlapping Cameras with Complex Motion (OCCM): Several cameras with overlapping views observe a relatively small region concurrently, and the motions

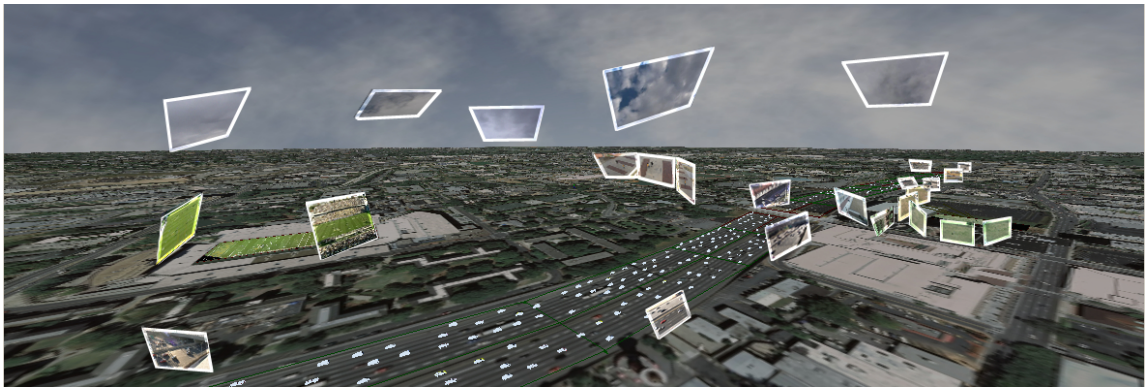


Figure 4: An overview of the "Augmented Earth Map": We make use of 36 videos to add dynamic information to city visualization. Dynamism in each input video of traffic, people, and clouds is extracted and then mapped onto the Earth map in real-time. Each video patch in the figure represents an input source.

within the area are complex. We demonstrate this scenario with people playing sports, which can be considered valid for other CCTV domains.

#3 Sparse Cameras with Simple Motions (SCSM): Sparsely-distributed cameras cover a wide area, but dynamic information is simple. For example, with independent observations of vehicles from different traffic cameras along a highway, we can predict the motions of the vehicles in nearby regions relatively easily.

#4 Sparse Cameras and Complex Motion (SCCM): Sparsely-distributed cameras observe different parts of a larger area, but the motions of target objects are complex. This case is one in which we observe and model natural phenomena such as clouds.

Fig. 4 presents an overview of the augmented aerial map with additional dynamic information extracted from various sparsely-distributed video cameras with various conditions/configurations.

5.1 Direct visualization using fixed cameras

First scenario is where we capture a video from a single viewpoint and we are interested in projecting the view and the related motions onto an aerial view from an AEM. Specifically, we put virtual characters onto the AEM to visualize pedestrians.

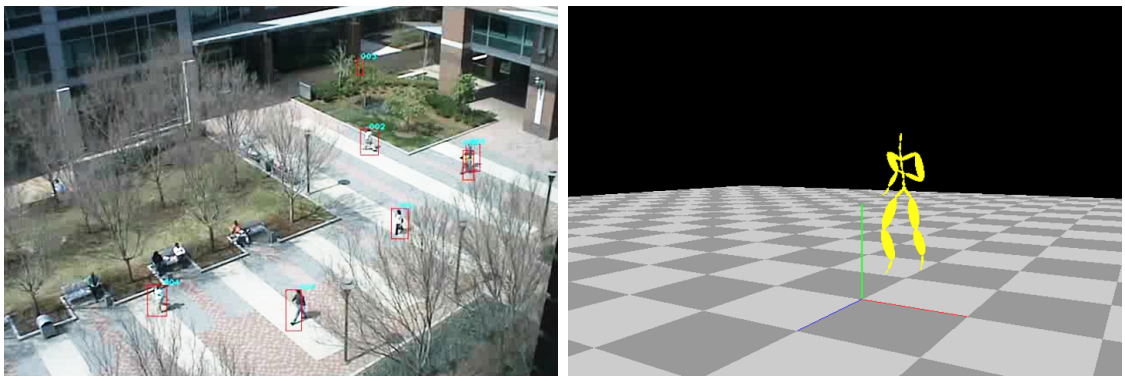
This scenario requires direct mapping (DM) of tracked objects in a video frame onto the virtual plane. Direct mapping is a visualization only from a data extracted from a single camera, hence we only visualize event within the camera.

This is the basic scenario and an essential building block for all of the other scenario cases described at the beginning of Chapter 5. Our approaches here build on previous efforts aimed at surveillance applications [43, 113, 116, 143].

We rely on direct mapping to visualize pedestrians in videos. As shown in Fig. 5, we first track the pedestrians and extract screen-space coordinates and velocities. These measurements are directly registered onto the plane space of the virtual environment. If the homogeneous coordinates in a video frame are $\mathbf{p_{x,y}} = [\mathbf{x}, \mathbf{y}, \mathbf{1}]^{\mathbf{T}}$, the new 2D location at planar space on virtual environment $\hat{\mathbf{p}}$ is simply calculated by $\hat{\mathbf{p}} = \mathbf{H}\mathbf{p_{x,y}}$. Subsequently, if the objects (pedestrians) are moving in the video with the velocity $\mathbf{v}$, we can also project the velocity onto the earth map plane by $\hat{\mathbf{v}} = \mathbf{H}\mathbf{v_{x,y}}$. This velocity is used to match simple motion data gathered off-line to visualize moving pedestrians. We used motion capture data from [90] to generate similar character animations for our stand-in avatars. We sample the trajectories of objects, then insert exactly one cycle of walking data onto them and interpolate the positions.

There is no accounting for visualizing movements that are outside the camera view and we are dependent on tracking approaches. This can be problematic when we have crowds in the scene and tracking fails due to occlusion. Some of these issues are addressed in the other scenarios, discussed later.

Note that currently, we are not interested in classifying objects or recognizing their states in the scene, which is beyond the scope of this work. In this scenario, we assume moving objects on a sidewalk are only pedestrians and they are engaged in simple walking motion and other similarly simple linear actions. On the road, we can assume the object of interest is a car.



(a) (b)

Figure 5: Example of Direct Mapping: (a) Pedestrians are tracked from single video (b)Tracked pedestrians are matched to motion capture data, then mapped onto virtual plane space, then rendered only within a coverage of the given field of view.

In summary, DM can be used when (1) a region of interest covered by a single view point (2) the motions of objects have to be simple. In Sections **??**, we will introduce methods to handle more complex situation in different scenarios.

5.2 Overlapping Cameras, Complex Motions: Sports

We now move to the domain where we have overlapping views and motions that have some structure, but there are several motions at the same time. While we have employed this case for a variety of scenarios, we are going to demonstrate this in the domain of sports (other examples are also shown in video and in Fig.Survaillance example). Sport is an interesting domain as we usually do have multiple views and in most instances we can rely on the field markings to help with registration. The overlapping views in this domain also require additional types of modeling and synthesis beyond the direct mapping from a single view (Section 5.1). Particularly, we also need to employ techniques to support view blending as the viewpoints are changed from one to the other, as we visualize the AEMs.

Observations and Estimation: We start by obtaining field of views (FOVs) $\mathbf{f}_i$ and camera homographies $\mathbf{H}_i$ (from each view to a corresponding patch in the virtual

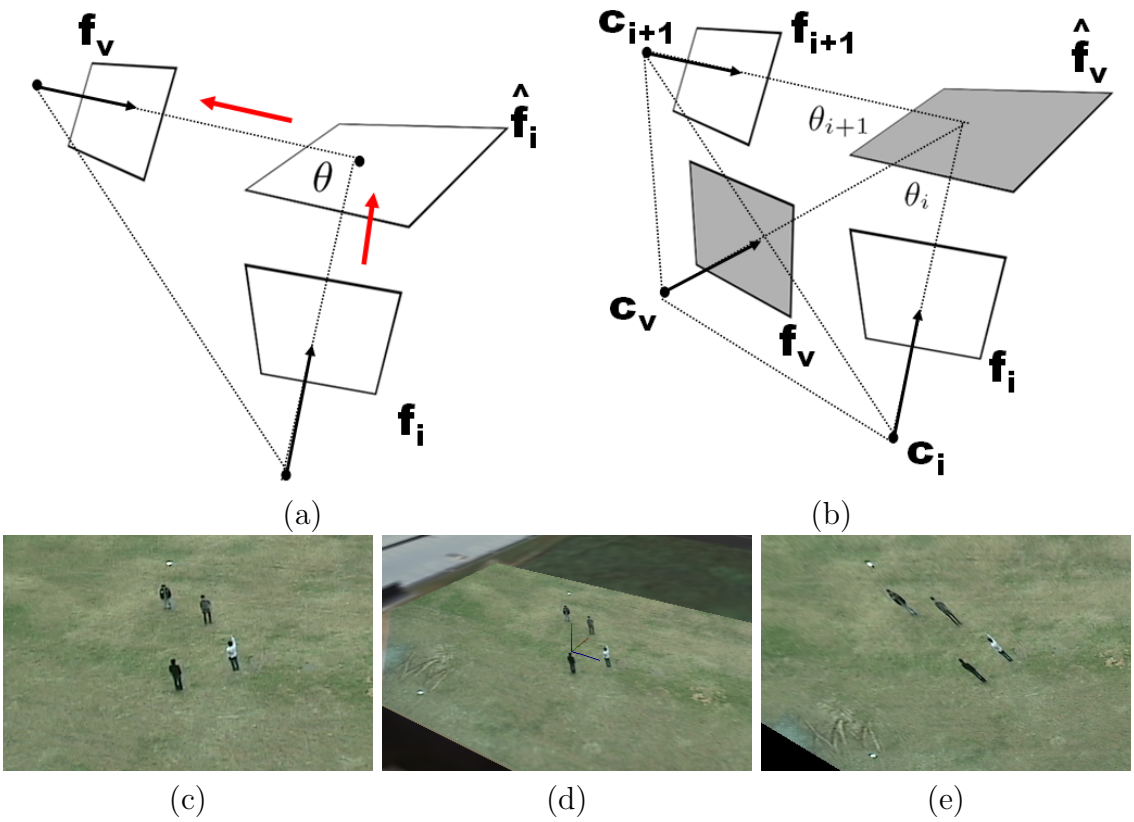


Figure 6: The relationship between each view:(a) Virtual view $\mathbf{f}_v$, Sample view $\mathbf{f}_i$ and the angle between them θ (b) Two consecutive views $\mathbf{f}_i$, $\mathbf{f}_{i+1}$, Back-projected virtual view $\mathbf{f}_v$ and blended rectified view $\hat{\mathbf{f}}_v$. Note that θ_i is an angle between $\mathbf{f}_i$ and $\mathbf{f}_v$ and θ_{i+1} is an angle between $\mathbf{f}_{i+1}$ and $\mathbf{f}_v$, (c) Source video having view $\mathbf{f}_i$(d) Back-projected virtual view $\mathbf{f}_v$ (e) Rectified view $\hat{\mathbf{f}}_v$: Note that (c) and (d) look almost similar view since the θ is almost zero.

plane) from the videos as described earlier. Then, the videos are rectified to top-views based on the extracted homographies, and registered onto the corresponding regions of the virtual earth environment. Additionally, we also calculate camera locations via calibration. We rely on the fact that (1) we have multiple videos, and (2) ground yard commonly provides good features, *e.g.*, lines and known distances on the field.

Once the videos are registered, the rectified top views are used as a texture on the AEM plane. Then, this textured video is re-projected to a virtual view based on the model view matrix in the AEM environment. We refer to this view as *Back-projected view* and the angle between the original and the back-projected views as θ. Fig. 6(a)

(a) (b)

Figure 7: The range of approximately invariant to distortion: (a) and (b) both are back-projected scenes from same video (Notice that within small change of viewing angle it still looks reasonable).

shows the relationship between back-projected virtual view $\mathbf{f}_v$, rectified view $\hat{\mathbf{f}}_i$ and original video $\mathbf{f}_i$.

View Synthesis and Blending: Now, we address the issue of view synthesis as we move from one view to another within the AEM environment. Past approaches use billboard style methods that change the orientation of the original view so that it always look at the virtual view [143]. While commonly used in practice, this approach only shows rough segment of video streams at a given 3D environment. To visualize sports games, LiberoVision system [85] uses segmentation of players and foreground warping. But it requires extensive batch-process time to make realistic view transition and only used in static scene. Seitz [118] proposed view morphing methods where they reconstruct virtual views from the a pair of images under the assumption that the cameras have well-calibrated epipolar geometry. This can be done when we have precise stereo pairs and have exact pair sets of local correspondence. Thus, it is not the case for us where the camera center of the virtual views is often out of the epipolar line and finding the correspondences at every frame is computationally demanding (if not impossible).

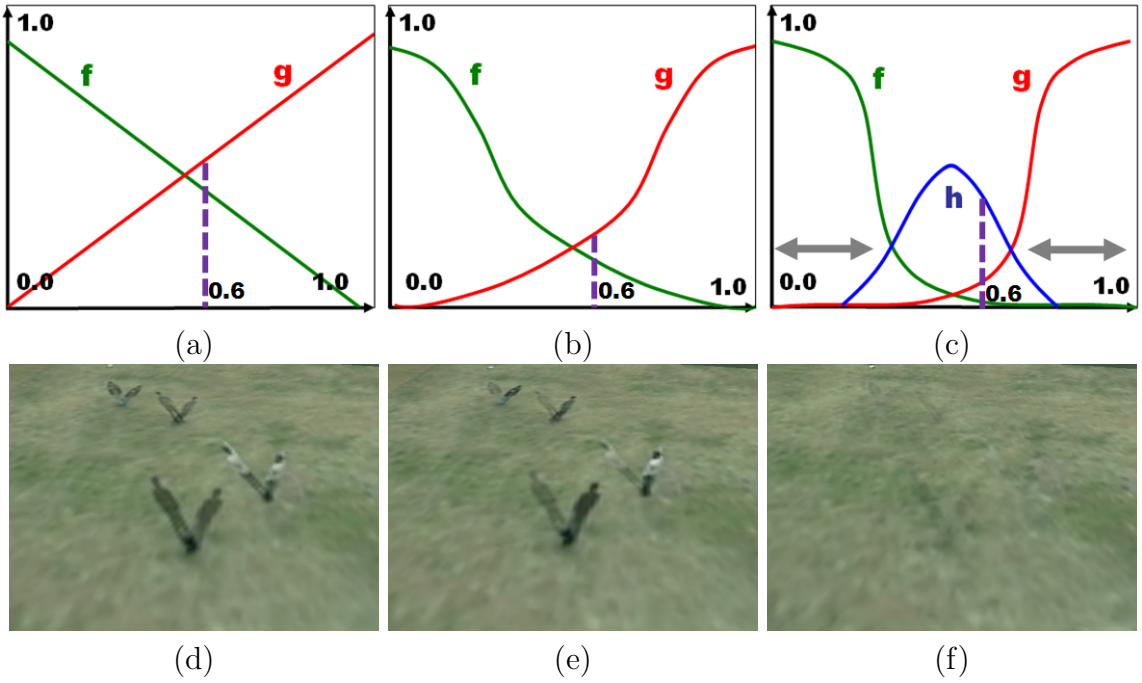


Figure 8: View blending with different interpolations, where ω_i is 0.6 :(a) Linear (b) Bicubic (c) Bicubic and Background blending: f is $f(\omega_i)$, g is $g(\omega_{i+1})$ and h is ω_{bkg} which is a weight for subtracted background image. (d),(e) and (f) show the back-projected views based on each interpolation. The gray arrows in (c) show the ranges of ω where the distortion is minimized, as shown in Fig. 7(a)(b).

We propose a more flexible global view blending method that can incorporate views and their subtracted backgrounds with overlapping regions and does not require an extensive pre/post processing. This approach is adequate for the scenario of our system in which the rapid registration of crowd-sourced videos is needed.

Once multiple views covering the same region are registered onto the AEM plane, their rectified views also overlap. Our goal is to generate virtual views based on the two consecutive views that exhibit the most similar viewing angle θ. First, we search for the pair of the closest two views exhibiting small θ's. Let these consecutive views and the corresponding angles be denoted by $\mathbf{f}_i$, $\mathbf{f}_{i+1}$ and θ_i, θ_{i+1} respectively(Fig. 6(b)). Then, we compute the two blending weights for both views based on the angle differences giving larger weight to the smaller angle: $\omega_i = \frac{\theta_{i+1}}{\theta_i+\theta_{i+1}}$ and $\omega_{i+1} = \frac{\theta_i}{\theta_i+\theta_{i+1}}$. If the ω_i is close to one, which indicates that the angle θ_i becomes zero, so that the viewing

vector of virtual view and the given view $\mathbf{f}_i$ are almost identical. This is the case where the virtual view is almost similar to $\mathbf{f}_i$, so that the virtually back-projected scene seems approximately invariant to distortions (See Fig. 6(c)(d)). Even when ω_i is less than 1.0, the distortion of the back-projected view is still negligible within some ranges (noted as gray arrows in Fig. 8(c)). This example is shown in Fig. 7.

In case we can not find a very similar view, we use interpolation and blending approach based on the weights computed from the angles between two adjacent views. In blending approach, if we blend $\mathbf{f}_i$, $\mathbf{f}_{i+1}$ using a linear interpolation, the blended region near the moving objects suffer from the ghosting artifacts, especially when both ω_i and ω_{i+1} become near 0.5, see Fig.8(d) for an example. This is because the virtual view $\mathbf{f}_v$ is placed out of range from both $\mathbf{f}_{i+1}$ and $\mathbf{f}_i$, and the camera center is out of the line $\overleftrightarrow{\mathbf{C_i C_{i+1}}}$ in Fig. 6(b). Even if we use bi-cubic interpolation, the same problem occurs (Fig.8(e)) although view transition becomes smoother.

In our solution, we blend not only a pair of rectified scenes ($\hat{\mathbf{f}}_i$,$\hat{\mathbf{f}}_{i+1}$) but also the subtracted background scenes generated from a pair of view videos. Now, suppose that $f(t)$ and $g(t)$ is a bicubic function for both views as shown in Fig. 8(c). Then, we blend each pixel in the virtual view as follows:

$$\mathbf{p}_v = f(\omega_i)\mathbf{p}_i + g(\omega_i)\mathbf{p}_{i+1} + \omega_{\text{bkg}}\mathbf{p}_{\text{bkg}} \tag{11}$$

where $\omega_{\text{bkg}} = 1 - (f(\omega_i) + g(\omega_{i+1}))$, $\mathbf{p}_{\text{bkg}} = \omega_i \mathbf{p}_i^{\text{bkg}} + \omega_{i+1}\mathbf{p}_{i+1}^{\text{bkg}}$, and $\mathbf{p}_i^{\text{bkg}}$ and $\mathbf{p}_{i+1}^{\text{bkg}}$ are the pixels of the subtracted background computed separately from $\hat{\mathbf{f}}_i$ and $\hat{\mathbf{f}}_{i+1}$ respectively (See Fig. 8(c)).

Now, the virtual view is almost identical to the background if the target view is out-of range from the both views. If the target view approaches a view to some extent, the synthesized view smoothly transit to the back-projected view of the closest viewpoint. Examples of smooth transition according to the change of virtual view point is shown in the video. In Section 5.3 we introduce an approach to visualize moving objects when multiple videos are not overlapped and each camera is distributed

sparsely.

5.3 Sparse Cameras with Simple Motion: Traffic

We demonstrate this scenario in the setting of analyzing videos of traffic and synthesizing traffic movements dynamically on AEMs. We are working in this domain following the ever-growing number of traffic cameras being installed all over the world for traffic monitoring. While the videos we are using are not recorded by actual traffic cameras installed by a department of transportation or the city government, our views are practically similar. The biggest technical challenge with this scenario is that as we have sparsely distributed, non-overlapping cameras, we do not have the advantage of knowing how the geometry or the movement from each camera is related to the other. While the topology of the camera networks still needs to be specified by user input, we do want to undertake the analysis of both registration of views and movement as automatically as possible.

To deal with the fact that we do not have any measurements in-between the camera views, we also need to model the movements in each view in general and connect the observations between cameras, i.e. to model the flow from one view to another. To achieve this, in our framework, we add two functionalities. (1) Modeling the flow using a graph-based representation. This allows to use the information obtained from observable regions and propagate it to infer a plausible traffic-flow across unobservable regions. (2) Develop a synthesis framework that can be driven from such data. We employ a behavior-based animation approach to synthesize flow across view and onto the AEM. Fig. 9(a) shows the topology of traffic nodes. Each node is a patch of the road and has a traffic state X_i and a synthesized state Z_i. A measurement Y_i is defined only in observable nodes monitored by videos.

Observation and Measurements: Using the tracking approach mentioned in Section 3.1.2, we can get estimates of position and velocities of objects from the video.

The measurement of i-th node Y_i consists of the positions and the velocities of the low-level motion features and the detected cars in the corresponding video frames. Once the low-level features within an observable node i are obtained, they are merged to form a set of objects $\mathbf{r}_i = \{r_{i,j}\}\forall j$ – cars, with j denoting an index of each car, based on the similarity and the connectivity formed from the features using deterministic approach [103, 135]. Experimental results demonstrate that it is sufficient for visualization and meet the overall purpose of the system. In summary, the measurement is defined to be the pair of the information on the low-level features $\mathbf{f}_i$ and the detected cars $\mathbf{r}_i$: $Y_i = \{\mathbf{f}_i, \mathbf{r}_i\}$.

The entire traffic system is also denoted as $\mathbf{X} = \{X_i | 1 \leq i \leq k_{\mathbf{X}}\}$ where $k_{\mathbf{X}}$ is the number of nodes. Additionally, the physical length of every i-th node is obtained from the available geo-spatial database and is denoted by d_i. Once the traffic system is decomposed into a graph manually, a set of *observable* regions $\mathbf{M}(\mathbf{X}) \subset \mathbf{X}$ with available video measurements are identified. On the other hand, the *unobservable* regions are denoted by $\tilde{\mathbf{M}}(\mathbf{X})$ where $\mathbf{X} = \mathbf{M}(\mathbf{X}) \cup \tilde{\mathbf{M}}(\mathbf{X})$. Note that every measurement Y_i is a set of low-level information computed from a video, rather than being the raw video itself.

The state X_i is designed to capture the essential information necessary to visualize traffic flow: (1) the rate of traffic flow n_i, and (2) the average velocity v_i of cars, *i*.e., $X_i = (n_i, v_i)$. By average flow, we mean the average number of cars passing through a region in unit time, whereas v_i denotes the average velocity of the cars.

The obtained measurement information Y_i is used to estimate an observable state $X_i \in \mathbf{M}(\mathbf{X})$, after the projection onto the virtual map plane using the available homography $\mathbf{H}_i$ for that region. First, the average speed $\hat{v}_i$ of the cars is estimated as an average of projected speeds of the low-level features w.r.t. the homography $\mathbf{H}_i$. Secondly, the flow of the cars passing through the ith region, $\hat{n}_i$, can be computed using the fact that the number of cars in the region $N_{\mathbf{r}_i}$ is the product of the flow

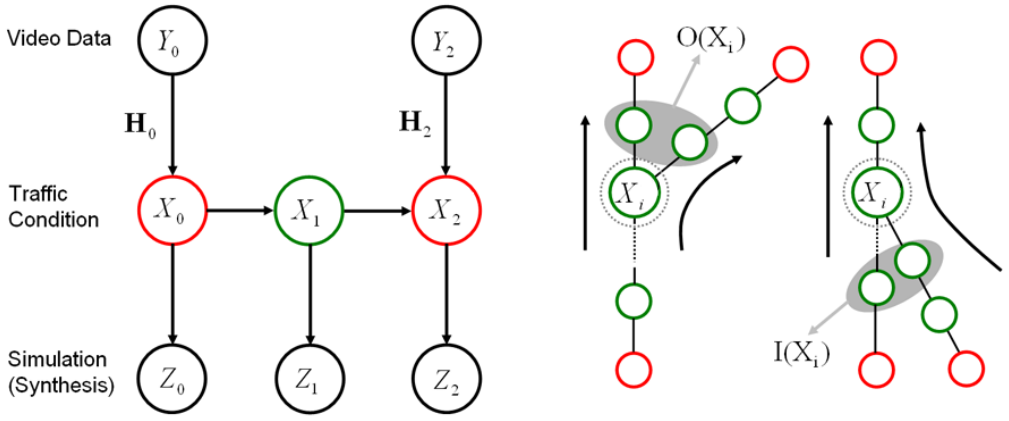


Figure 9: Graph-based representation of traffic nodes: Red nodes indicate observable region $\mathbf{M}(\mathbf{X})$ and Green nodes are unobserved regions $\tilde{\mathbf{M}}(\mathbf{X})$. (a) The middle chain corresponds to the traffic conditions on the graph which represents the traffic system. Node index i is 0 to 2 in this example (b) split: outgoing regions $\mathbf{O}(\mathbf{X_i})$ from node X_i are marked. (c) merging: incoming regions $\mathbf{I}(\mathbf{X_i})$ to node X_i are marked.

multiplied by the average time d_i/v_i for cars to pass a region, *i*.e., $N_{\mathbf{r}_i} = \hat{n}_i \cdot (d_i/\hat{v}_i)$.

Modeling from Data. Spatial Correlation to Infer Missing Data: Once the set of states $\mathbf{M}(\hat{\mathbf{X}})$ are estimated for the observable regions, they are used to estimate the unobserved states $\tilde{\mathbf{M}}(\mathbf{X})$. We adopt the Bayesian networks [99] formalism to exploit the fact that the unknown traffic conditions $\tilde{\mathbf{M}}(\mathbf{X})$ can be estimated by

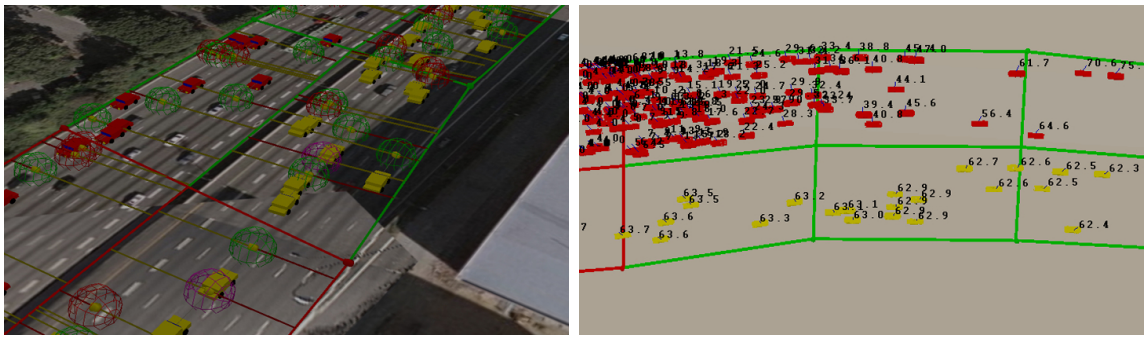


Figure 10: Simulated cars: (a) Each object(car) acts as an autonomous agent controlled by (1) local reaction based on their own perception range, (2) global dynamics extracted from video (b) Cars' reaction when average speed of upper-left corner is forced to be very slow, while other corners are controlled by real data.

propagating the observed information from the spatial correlation models. The whole traffic graph is a directed graph where an edge from a region X_j to another region X_i exists whenever traffic can move from X_j to X_i. For every node X_i, a local spatial model $P(X_i|\mathbf{I}(X_i))$ between the node X_i and the incoming nodes $\mathbf{I}(X_i)$ is specified (See Fig. 9). Once all the local spatial models are defined, the posterior traffic conditions $P(X_i|\mathbf{M}(\hat{X}_i)), \forall X_i \in \tilde{\mathbf{M}}(\mathbf{X})$ at the unobserved nodes are inferred using belief propagation [41, 99].

In detail, we make an assumption that the average flow $X_i|_n$ of the cars in a region X_i matches the sum of the average flow of the cars from the incoming regions $\mathbf{I}(X_i)$ with a slight variation w_i which follows white Gaussian noise : $X_i|_n = \sum_{j \in \mathbf{I}(X_i)} X_j|_n + w_i$. For velocity, we assume that the average speed in a region matches the average speed of the cars with a slight variation q_i which is again a white Gaussian noise : $X_i|_v = (\sum_{j \in \mathbf{I}(X_i)} X_j|_v)/N_{\mathbf{I}(X_j)} + q_i$. The variance of the Gaussian noises, both w_i and q_i, are set to be proportional to the length d_i of the target region i.

The above assumptions are applied in case the road configuration is straight or merging (see Fig. 9(a,c)). For the case of splitting (see Fig. 9(b)) , we make a different assumption on the dependencies for the flow $X_i|_n$. Let us denote the outgoing regions (children) of a node X_i by $\mathbf{O}(X_i)$. To maintain a tractable solution for the overall inference phase using belief propagation, the dependencies $P(X_j|X_i)$ between the parent node X_i and every child node $X_j \in \mathbf{O}(X_i)$ are encoded individually in a Gaussian form. The new assumption is that there is a known ratio on the traffic portion $0 \leq \alpha_j \leq 1$ ($\sum_{\forall j} \alpha_j = 1$) flowing from the parent node X_i to an outgoing node X_j. Then, the individual functional dependency is modeled as follows : $X_j|_n = \alpha_j X_i|_n + u_j$ where a slight white Gaussian noise is denoted by u_j, which is again proportional to the region length d_j. Note that the flow estimate at an outgoing region $X_j|_n$ is estimated based on not only the propagated information $\alpha_j X_i$ from its parent, but also on the information propagated back from its descendants in the

directed traffic region graph.

The posteriors $P(X_i|\mathbf{M}(\hat{X}_i))$ for the unobserved regions are concise Gaussians as all the spatial correlations are Gaussians. In the simplest case, *e*.g., straight configurations, the maxima (mean) of the resulting posterior estimates $\hat{X}_i$ is analogous to the linear interpolation of the nearest observable regions. For other configurations, the posteriors present more complex forms due to the interaction between multiple regions.

Synthesis. Parameterized Traffic Simulation: To visualize traffic flow based on the estimated traffic states $\hat{\mathbf{X}}$, we developed a parameterized version of Reynolds' behavior simulation approach [111] where we adopted the steering behavior [112] for our implementation. By parameterized behavior simulation, we mean that the cars are controlled by the associated behavior-based controller, but the behaviors are parameterized by the current traffic condition. The controller of a car in the i-th region is parameterized by X_i. Hence, the behavior of a car varies if the estimated traffic condition within a region changes or if the car moves onto adjacent traffic regions.

The flock simulation status within the i-th region is denoted by $Z_i = \{s_{i,k}\}_{k=1}^{N_{Z_i}}$ which comprises of a set of simulated cars $s_{i,k}$ with corresponding position $s_{i,k}|_p$ and velocity $s_{i,k}|_v$ attributes, where N_{Z_i} denotes the total number of cars. The synthesis of the flock status Z_i^t at time t should be consistent with both the already existing cars from the past scene Z_i^{t-1} and the newly added cars coming in from the virtual scenes corresponding to the incoming regions $\{Z_{i\in\mathbf{I}(j)}^{t-1}\}$. In terms of the adjustments of the speed $s_{i,k}|_v$ between the frames, the cars adjust their speeds to meet the estimated average speed of the region $\hat{X}_i|_v$ where their accelerations are set in such a way that the speeds are approximately linearly interpolated within the region. Fig. 10 shows an example of successful simulation in (a) normal and (b) heavily loaded traffic.

5.4 Sparse Cameras with Complex Motion: Clouds

Our final scenario is aimed at using videos of natural phenomenon, primarily clouds and adding them to AEMs for an additional sense of reality. This is a hardest case, since clouds consist of particles, thus it does not have a specific shape. Moreover, the coverage of field of view from each observing camera is relatively small compared to the region displayed as sky. In this scenario, we use video of moving clouds to extract procedural models of motion, which are then interpolated and rendered onto the AEM, with a compelling effect. Some of the AEMs are now moving towards rendering clouds from satellite imagery and our work is similar to their goals, but we use sparse video and use procedural modeling to generate the clouds, rather than playback what was captured directly.

Observation and Extraction: In this scenario, cameras are spatially distributed and only a small subset of the targeted sky area that is to be synthesized is visible by the FOVs of the cameras. The entire sky scene is synthesized by: (1) the extracted local clouds density within the observed regions based on the video, (2) the interpolated cloud density information within the unobserved regions from the Radial basis function (RBF) technique, and (3) entire cloud imagery synthesis based on the cloud template obtained by procedural synthesis techniques and the computed cloud density map. For measuring dynamic movement of clouds, we also extract velocities from videos. We assume that the video used to extract velocity is always taken by camera having 90 degree of elevation, and zero degree azimuth. We call this video an *anchor video*.

Cloud density interpolation from multiple sky videos: We use a Radial Basis Function(RBF) [19] to globally interpolate density of clouds in unobserved sky region based on multiple videos. The main concept of our interpolation follows a method

described in Turk and O'Brien [131]. They interpolate an implicit surface from a given set of scattered data points. We use this method to interpolate density of unobservable region in the sky using densities extracted from given set of clouds videos. In our work, constraint points are the location of feature points(x_i, y_i) extracted from each input video, and the basis vector is defined as $\mathbf{G} = [G_1(x_1, y_1), \ldots, G_n(x_n, y_n)]^\top$ encoded by strong gradient and velocity vectors representing density of clouds. Now a basis function d_{ij} between any constraints points is chosen as $||(x_i - x_j)^2 + (y_i - y_j)^2||$.

Using these measurements we can globally interpolate the cloud density of any points in unobserved sky region by weighted sum of basis function. The weights for

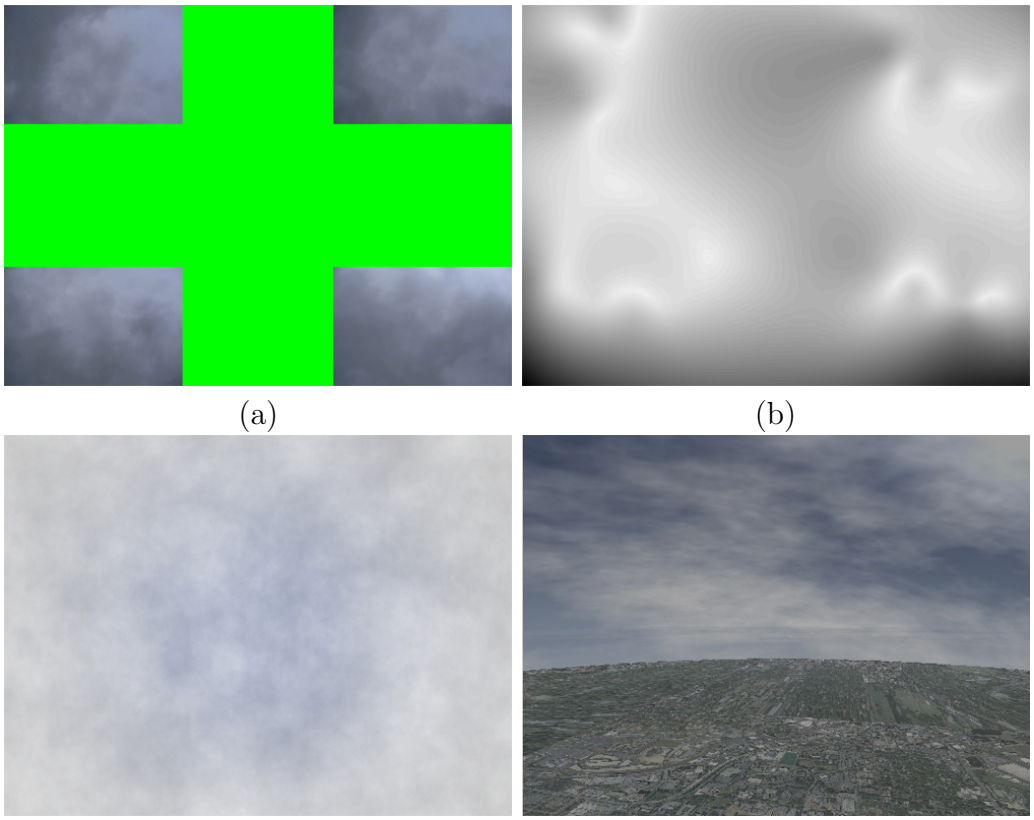


Figure 11: Generating clouds layers procedurally using videos: (a) 4 input videos (Green region is an unobserved region)(b) Globally interpolated density map from RBF (c) result cloud layer (d) registered onto the sky dome in Earth Map environment.

the interpolated density is obtained by solving following linear system:

$$\begin{bmatrix} d_{11} & \cdots & d_{1n} & 1 & x_1 & y_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ d_{n1} & & d_{nn} & 1 & x_n & y_n \\ 1 & \cdots & 1 & 0 & 0 & 0 \\ x_1 & \cdots & x_n & 0 & 0 & 0 \\ y_1 & \cdots & y_n & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} G_1 \\ \vdots \\ G_n \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12}$$

where, the λ_i are weights, and c_i is a degree one polynomial that acounts for the linear and constant portions of density. More details of underlying basis for RBF interpolation is described at [131].

Now, the generated density map is used as a density function for procedural modeling of cloud textures. Figure 11(a)(b) shows an example of density map generated from four videos. However, if the generated density map is not appropriate due to mis-detection of feature vectors, the system provides an user interface to edit the density map by adding additional basis vectors.

Sythesis: Procedurally generated clouds A procedural texture is a synthetic image generated based on random functions [81, 101]. We used the 2D version of the approach suggested by [88] to synthesize a 2D cloud template. In detail, the function f below is used to generate a cloud template from a set of weighted sub-templates computed at multiple scales using a random number generator g applied with seeds which increase the with the details each particular template is targeted to produce:

$$f(x,y) \;=\; |\sum_{i=0}^{K-1} P^i \cdot g(2^i \cdot x, 2^i \cdot y)| \text{ where } P \subset [0,1]$$

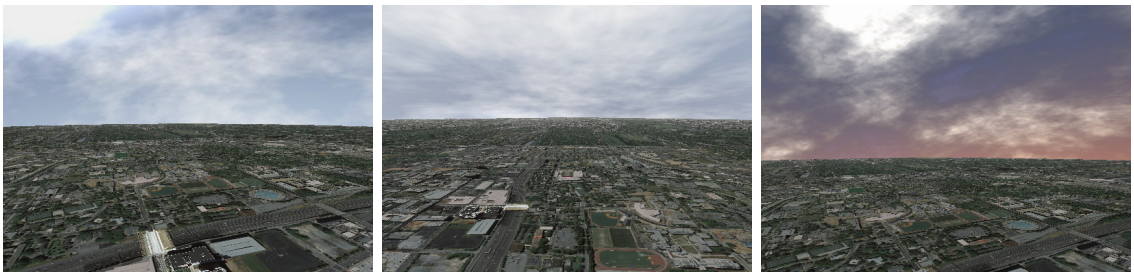
where K is the number of scales and P denotes the persistence constant. Note that the sub-template computed for the lower-scale with smoother characteristics are weighted more heavily than the sub-templates for the higher-scales. While there are

many options for the random number generator g, we used the function suggested by [88].

We also generate multiple cloud templates with different persistent levels and seeds for realistic sky movement. Once cloud textures are generated, we make additional layers having different effects to generate realistic sky animation. Similar approaches are in use for real-time sky rendering systems [49, 142]. The sun mask and glow mask is generated based on the time (*i*.e., At noon, sun would be appeared on the center of texture). A sky map is used to make color of sky to be brighter if the location is near the ground. Then, the finally generated sky textures are mapped onto the sky domes [152]. To visualize sky, representing a dynamic of current sky, the mapped sky textures moves based on the velocity captured from anchor video.

5.5 Discussion of Experiments and Results

To validate our approaches, we undertook a series of experiments for different scenarios on a variety of challenging domains, under varying conditions. For the direct mapping scenario, cars and pedestrians are animated based on single videos where we used available motion capture data to animate individual pedestrians. For the view blending examples of sports and other instances, videos are collected from three different testing locations where a total of 10 cameras were used. For the football



(a) (b) (c)

Figure 12: Various results of sky rendering based on multiple videos at different times.

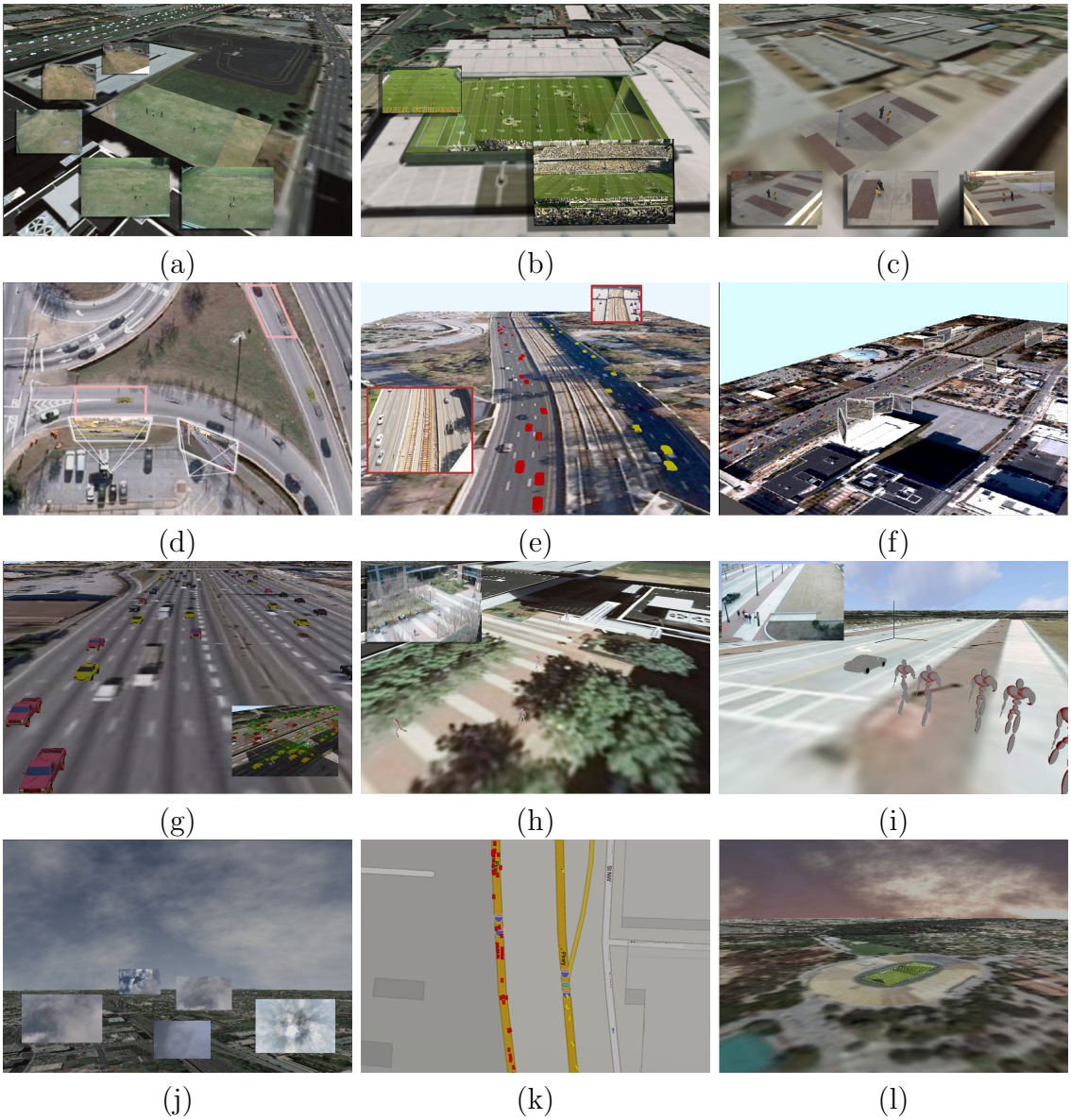


Figure 13: Results from our prototype system using 36 videos: 1. OCCM (View Blending): (a) 5 Cameras for soccer game (b) Two broadcasting footages of NCAA Football game (c) Three surveillance cameras. **2. SCSM (Traffic):** (d) Merging Lanes (e) The steering behaviors of cars are demonstrated in the slightly curved way (f) 8 cameras for larger scale traffic simulation including merge and split (g) Rendered traffic Scene and corresponding simulated scene **3. DM (Pedestrians):** (h) Direct mapping of pedestrian having simple motion (i) Visualization of pedestrians and cars in the street **4. SCCM (Clouds):** (j) Four videos for clouds and sky generation **5. Customized visualization using user interface:** (k) Traffic flow are visualized in a line map by adjusting the scale of moving objects (l) The game scenes can be augmented into different stadium. Please see the video on the project web site.

Figure 14: Qualitative evaluation of traffic in different lighting and weather conditions Rendered Dynamic scenes based on **(Top)** video data captured at night : Resulting scene demonstrates sparse traffic distribution with high velocity across almost entire nodes, **(Bottom)** video data captured during snowing weather condition : Despite heavy noises in the video due to snow, the result shows flows that realistically demonstrate real-world traffic; one lane (marked as red) shows densely poppulated heavy traffic while the other lane (yellow) is relatively sparser.

game visualization in the stadium we used the video footage provided by the Georgia Tech Athletic Association (made available for research purposes). For applications of traffic visualization, 6 different data sets are collected with number of video feeds varying from 2 to 8 each, where 3 data sets were captured at an identical site but at different time and under different weather conditions. Finally, for the cloud visualization, we collected four different data sets where 3 to 6 cameras were used each. For each cloud data set, a separate anchor video was captured to measure the velocity of

clouds. Since a variety of the above experiments are showcased in the enclosed video, we would briefly highlight the results in this section.

The prototype system is developed using C++/OpenGL on a computer with Quad-core 2.5GHz, 2GB RAM, and NVidia Quadro FX770M graphics card. The resulting visualizations are rendered in real-time at approximately 20 frames per second where 500 targets can be tracked at maximum for the traffic flow scenario.

Through our experiments, the scenario (Fig. 13(a,b,c)) that requires view and background blending demonstrates view transitions that are smooth and provides dynamic visualizations.

In the traffic scenario, the visualized traffic closely matches the average speeds of the real traffic system. However, it was noted that the quality of the estimated traffic flow deteriorates in proportion to the distance between the cameras. The cameras used in our results are placed no more than 0.5 miles away, and provide qualitatively plausible visualizations. Nonetheless, it can be observed that our results show fairly accurate *global tendency* which reflects real-world average flows (velocities). However, it is important to note that it does not guarantee the exact individual position of every car in unobserved regions because the positions of cars in the unobserved region are designed to be driven by behavioral model where target velocities are computed through interpolation of observed ones.

To evaluate the parameterized flock behavior, we artificially forced a very slow speed only to the left-top observable node, as already shown in Fig. 10(b), while the other nodes still received live information, to mimic a traffic congestion. The objects approaching the congested region changed their velocity sharply but realistically under the poised behavioral rules.

Our system produces reliable results under diverse lighting and weather conditions for traffic visualization (Fig. 14). In terms of the test results on a challenging road topology, Fig. 13(d,e) depicts a scene with curved roads that merge into a speedy

highway. The cars simulated based on the live videos successfully adjust their speed on the curved roads (approximated by a series of rectangular regions) after which they accelerate into the highway to catch up with the average speed.

In the challenging large-scale experiments shown in Fig. 13(f), 8 cameras are installed at different locations where the visualized road system consists of 13 observed and 26 unobserved regions. Some cameras observe one-way road while others observe two-way and the traffic topology include merge, exits and bridge crossings.

The results for the DM scenario (Fig. 12(h,i)) demonstrate that pedestrians and cars are animated properly based on the successful tracking results.

Various styles of sky and clouds are generated as shown in Fig. 11 and Fig. 13(j). Although the visualization results do not capture the exact shape of the individual clouds or the exact sky atmosphere, movement and density of the distributed clouds reflect the characteristics of the input videos plausibly.

Finally, it is worth noting that the interface developed in our prototype system allows us to make the customized visualizations with ease. Fig. 13(k) demonstrates the traffic visualization on the line map (*i*.e.google map), and Fig. 13(l) shows an example where we map a football game onto a different stadium far away from the original location.

5.6 Summary

In this chapter, we introduced methods to augment Aerial Earth Maps (AEMs) with diverse types of real-time information, namely pedestrians, sports scenes, traffic flows and skies. The proposed set of solutions are targeted to address different types of scenes in terms of camera network configuration/density and the dynamism presented by the scenes. The prototype system which integrates all the components run in real-time and demonstrates that our work provides a novel, more vivid, and more engaging virtual environment through which the users would browse the cities of now. Our work

showcases a unique approach that provides improved experience for users to visually consume "what is happening where", to augment the conventional text-web-browsing experience currently provided by static AEMs.

It is important to note a few of our limitations. First, the direct mapping method only visualizes the object moving within its view, and movement modeling outside is only possible when overlapping views are available. Second, the tracking within the direct mapping and the traffic flow estimation assumes that the occlusion among targets is modest and the target association can be computed relatively accurately. Accordingly, none of our current approaches for tracking will work for dense crowds or for low-frequency videos (which are often the case for many surveillance cameras), due to significant occlusions. Third, the tracking algorithms used in our work are not able to recognize the detailed postures of complex targets, *e*.g., human postures. While efforts have been made to handle this task in computer vision (*e*.g., [35, 106]), practical performance still depends on specific constraints. Fourth, we cannot directly apply our traffic flow approaches to the scenes with higher-level controls and behaviors, *e*.g., intersections with traffic lights, which is an interesting avenue for future research. Finally, our solutions do not support modeling or analysis of high-level semantic information, *e*.g., car accidents or street burglaries.

We do note that the wide spread distribution of cameras, which are used by our approach, rightfully brings up serious privacy issues. This is due to the concerns of direct exposure of individuals (and their privacy) to massive distribution of public surveillance cameras [144]. *Street Views*, by Google has similarly raised privacy concerns because it sometimes displays people.

In our current work, we are intentionally "abstracting" individuals and our camera distance to individuals is large enough, that personal identity is hard to ascertain. We believe our approach of symbolizing moving objects by tracking without identification can alleviate the privacy concerns. However, we do see the concerns raised and propose

that the data sources and the entire system can be managed properly. In the present prototype system they are explicitly controlled.

It is also worthwhile to discuss a bit more on the potential of our system. *First*, OCCM application can be used as the interactive sport broadcasting where multiple broadcasting cameras or crowd-sourced videos incorporate to visualize the sport event on virtual environment. It will allow users to actively watch sports games by selecting favorite views and regions of interest. *Secondly*, new types of advertisement could be brought into our system. For instance, crowd events within the field of views of public cameras can be used as the novel types of marketing methods by immediate visualization in the virtual environment or map. *Third*, our approach provides visually fused information that integrates individual sensor observations, and is potentially useful for surveillance, security, and military applications. *Finally*, our system can be used as a mirror that one can interact between real worlds and virtual worlds. This is possible because *Augmented Reality* (AR) techniques allow us put the virtual objects onto the screen, which displays the real-world scene, while our approach puts the present dynamic information onto virtual world. Imagine your avatar in the virtual world application such as Second Life [74] looking at the moving objects that are actually reflecting the movements in the real world and vice versa. We believe such interactions can be an important milestone for the future of virtual reality as well as new types of cyber-cultures [121].

In summary, we have presented a novel prototype system to augment dynamic virtual cities based on real-world observations and spatio-temporal analysis of them. We feel that our approach opens doors for interesting forms of new augmented virtual realism.

In our future work, we aim to overcome the above limitations, and incorporate even more types of additional dynamic information such as river, swinging forest, sun, weather patterns, environmental condition and even aerial objects like birds and

airplanes. Additionally, we would like to apply our system onto various types of possible applications mentioned above.

CHAPTER VI

GLOBAL MOTION ANALYSIS FOR ADJUSTING THE VIEW POINTS OF A CAMERA

In this chapter, we focus on the analysis of complex dynamic scenes in which multiple objects are moving in a video. As already discussed in Chapter 2, analyzing and visualizing a dynamic scene are difficult, even with a pan-tilt-zoom (PTZ) camera observing the scene. In the following sections, we show that the SDI/SDA approach in a spatio-temporal domain can provide an effective way of optimally controlling the PTZ camera to automatically visualize complex dynamic scenes in several sports games. We first introduce a method of generating a dense motion field from the movement of each object using scattered data interpolation. The dense motion field eventually represents a global behavior from a set of players on the field. We then show how the generated global motion field can be used for predicting important future locations, and how the detected locations can be applied to adjusting the field of view of the PTZ camera.

6.1 Observation on complex dynamic sports scenes

Understanding complex dynamic scenes in team sports is a challenging problem. This is partly because an event in a game involves not only the local behaviors of individual players but also structural global movements of players. We are interested in automated analysis of such complex scenes with multi-agent activities, and consider that the tracking of multiple agents can be used to analyze these scenes, extract interesting events, and even predict what is likely to happen. We draw motivation from a quote by Wayne Gretsky, the legendary hockey player, *"A good hockey player plays where*

the puck is. A great hockey player plays where the puck is going to be." Our work is based on the assumption that the players themselves have the best view and clearest understanding of the development of a game during play. The players' movement on the field reflects their interpretation, and possibly their intent, based on their role in the game, which we should leverage for interpreting the state of the game.

Our hypothesis is that higher level information can be deduced by tracking and analyzing the players movements, not only individually but also as a group. In this study we describe a method to build a global flow field from players ground-level motions. We propose the novel concept that the flow on the ground reflects the intentions of the group of individual players based on the game context, and use this for understanding and estimating future events.

In this work, we specifically focus on the sport of soccer (i.e. football). For example, consider the soccer scene in Fig. 15, which demonstrates play evolution. The goalkeeper passes the ball to a nearby defender (top), but one of the offensive players sees an opportunity to intercept/steal the ball. One second later (bottom) we see the goalkeeper and another defender start moving to location 2 to prepare to respond to an offensive interception. The players are tracked on the ground plane to generate a flow field (shown by the white vector field) which in turn is used to infer possible locations of future events, noted by red circles.

Our primary contributions in this work are: (1) *Extracting ground-level motion* from individual players movement from multiple-views. (2) Generating a dense flow field from a sparse set of individual players motions, a *motion fields* on the ground. And, (3) Detecting the locations where the motion field converges and inferring the *play evolution*.

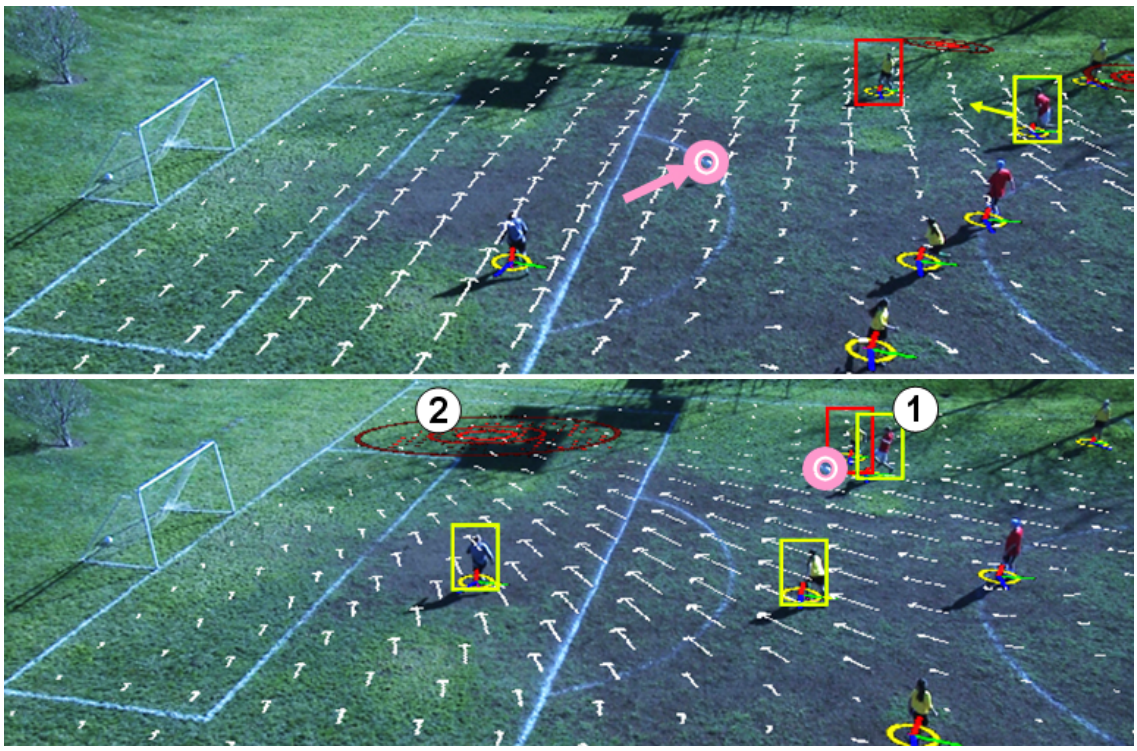


Figure 15: Examples of how players movement indicates play evolution in a dynamic sport scene: The motion field on the ground is denoted as white arrows, and the locations where play evolution is predicted are denoted as red iso-contours. Ball location is highlighted with a circle. **Top:** The goalkeeper passes the ball to the last defender (red box) while an offender (yellow box) is moving to intercept him. **Bottom:** One second (30 frames) later, at the moment of interception (position 1), the goalkeeper and another defender (yello boxes) are moving in the direction of position 2. This indicates the possible location of a future event.

6.2 Motion Field Construction from Video

The first step in constructing a motion field is extracting tracks of individual players on the field. While single camera tracking is possible for soccer [126], and could be useful for our approach, for this work, we have decided to rely on more robust multiple player tracking using multiple views.

View dependent analysis for player tracking using multiple camera suffers from a data fusion problem. In addition, flow analysis may be sensitive to the perspective distortion of different views. To address these issues, our approach is to analyze the game scenes from a *top-view* warped image of the ground plane. The top-view is

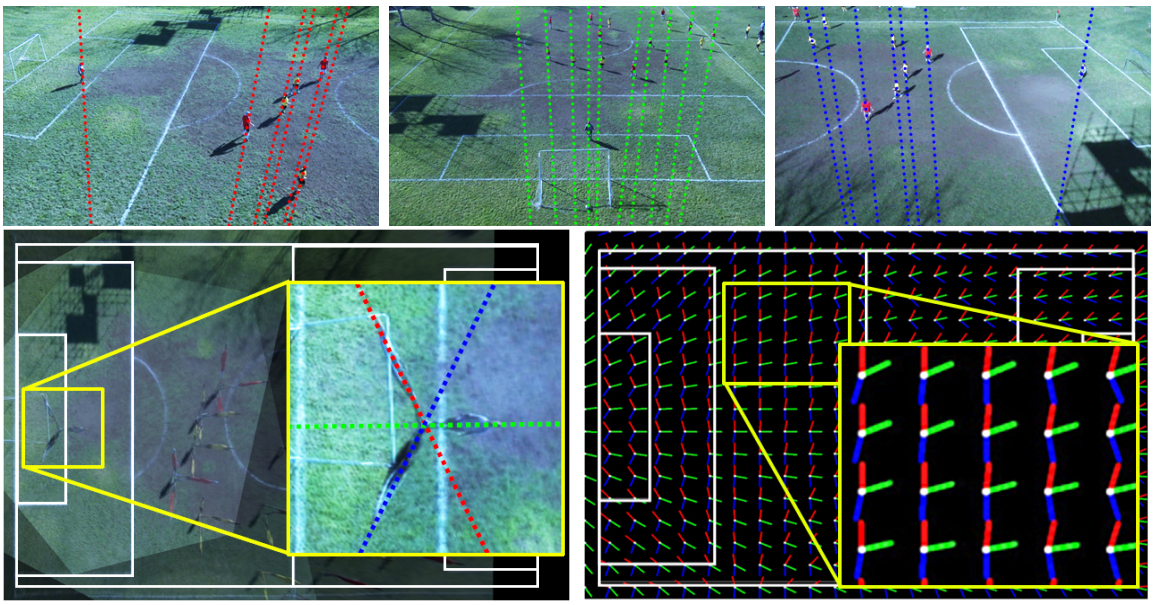
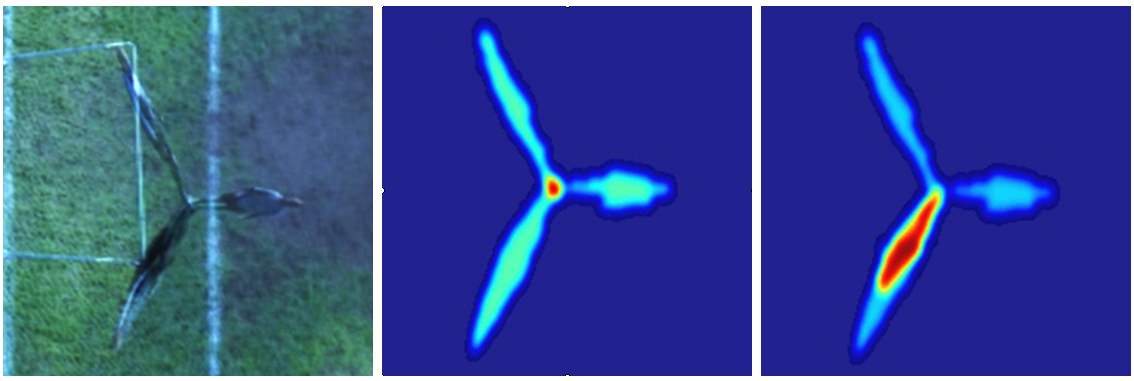


Figure 16: Geometric Constraints: (Upper row) Each view ($\mathbf{I}_k$) has a vertical vanishing points ($\mathbf{v}_k$). (Bottom left) In the top-view warped image ($\mathbf{I}_k^{top}$), players are distorted in the direction of the projected vanishing points ($\hat{\mathbf{v}}_k$) of each view. (Bottom right) The location of a player on the ground is identified by the intersection of these projections on the ground plane.

constructed by combining the warped footage of each of the multiple cameras. We then extract a multi-view consistent player location in the top-view by optimizing the geometric constraints shown in Fig. 16. This allows us to create the individual players' ground level motion. (Sec. 6.2.1).

Through spatial and temporal interpolation we combine the tracked player motions to create our *motion field on the ground-plane* (Sec. 6.2.2). Finally, we analyze this motion field to detect and localize important regions (Sec. 6.2.3).

We first define some notations. Assume that we have N cameras. Let $\mathbf{I}_k$ ($1 \leq k \leq N$) refer to a frame of each camera and $\mathbf{I}_k^{top}$ is a top-view image where each $\mathbf{I}_k$ is warped through the homography $\mathbf{H}_k^{top}$. Additionally, $\mathbf{x} \in \mathbf{I}^{top}$ denotes that $\mathbf{x}$ is in the coordinate space of a top-view (ground field).



(a) (b) (c)

Figure 17: Position Confidence in top-view: (a) Overlapped top-view of warped views from each angle (each view of a player is warped over the direction of vertical vanishing points)(b) Normalized probability of the summation of the warped foreground probabilities. Note that the region which is close to the ground has higher probability (c) In the presence of shadow.

6.2.1 Extracting Individual Ground-Level Motion

To construct our flow field, we first extract the ground-level motion of individual players. At each time t this motion is defined as the velocity vector $[u\ v]^T$ representing a player's movement on the ground at a 2D location $\mathbf{x}(x, y) \in \mathbf{I}^{top}$. To find the motion, our algorithm first detects the ground position (optimized location near the feet) of each player $\mathbf{x}$ at a given time t. Then, we search for a corresponding position in a previous frame at time $t - a$ (where $a > 1$ for stability and is usually set to 5 frames). The motion velocity at time t is the difference between these two positions. Note that the individual motion is reconstructed at each time separately and does not require explicit tracking since it is only used to construct our flow field.

To find the 2D location of players on the ground we make use of the fact that each view has its own vertical vanishing point (VVP) $\mathbf{v}_k$. We denote the projected VVP onto the ground view as $\hat{\mathbf{v}}_k = \mathbf{H}_k^{top}\mathbf{v}_k$ $(1 \leq k \leq N)$. Each $\hat{\mathbf{v}}_k$ gives us a unique direction in any location on the ground (see Fig. 16). Using background subtraction we define for each pixel in each view a confidence measure of that pixel being part of the foreground (i.e. player) or background (i.e. grass field) [77]. We

combine all measures from all views on the ground plane by summing their projections and normalizing to get the *position confidence* map $\mathbf{PC} : \mathbf{I}^{top} \rightarrow [0, 1]$, where $\mathbf{PC}(\mathbf{x})$ is the probability that $\mathbf{x} \in \mathbf{I}^{top}$ is part of the foreground (Fig. 17).

Since the region around each player's foot is located on the ground plane where the homographies for each view are extracted, the probability of foreground in those regions will be higher than in other regions (Fig. 17(b)) [64]. However, if there are cast shadows, the shadow region will also have high foreground probability (Fig. 17(c)). Therefore, we consider the highest **PC** position only as an initial location of the player. We define a window $\mathbf{W}_{init}$ around the initial position and refine it based on geometric constraints.

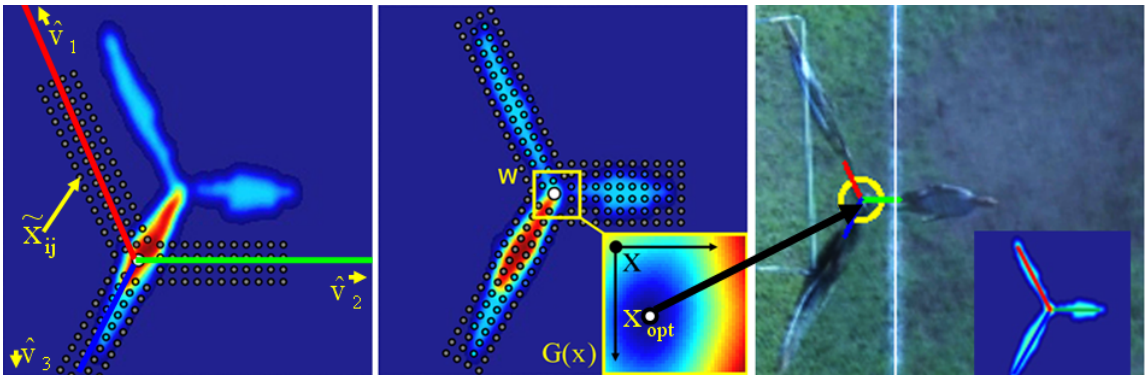
The geometric constraints are included by searching for the intersection point of the foreground projection of all N directions, where N is the number of views. This intersection is the weighted centroid of all foreground samples ($\mathbf{x}$ s.t. $\mathbf{PC}(\mathbf{x}) \neq 0$) along each projected VVP in all N directions (Fig. 18(b)). We define the player ground location cost function $\mathbf{G}(\mathbf{x})$ and search for its minimum inside $\mathbf{W}_{init}$. $\mathbf{G}(\mathbf{x})$ is the weighted summation of the distance between a set of foreground sample points $\tilde{\mathbf{x}}_{i,k}$ and a line axis established by $\mathbf{x} \in \mathbf{W}_{init}$ and each projected vertical vanishing point $\hat{\mathbf{v}}_k$,

$$\mathbf{G}(\mathbf{x}) = \sum_{k=0}^{N} \sum_{i=0}^{n_k} \mathbf{PC}(\tilde{\mathbf{x}}_{i,k}) \cdot \mathrm{d}\left(\tilde{\mathbf{x}}_{i,k}, (\hat{\mathbf{v}}_k - \mathbf{x})\right), \tag{13}$$

where n_k is the number of foreground samples based on each direction k, and we use $\mathbf{PC}(\tilde{\mathbf{x}}_{i,k})$, the probability of being foreground, as the weight for each of the foreground sample points.

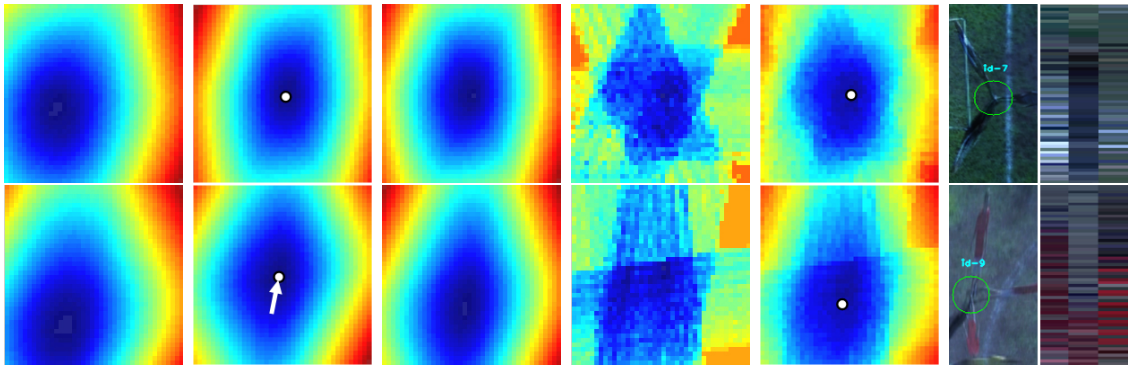
As shown in Fig. 18, the evaluation based on $\mathbf{G}(\mathbf{x})$ is performed over all directions simultaneously (in this case $N = 3$). The optimal ground-level position of the player is $\mathbf{x}_{opt} = \arg\min_{\mathbf{x} \in \mathbf{W}_{init}} \mathbf{G}(\mathbf{x})$.

Note that the set of sampling points $\tilde{\mathbf{x}}_{i,k}$ for each $\mathbf{x} \in \mathbf{W_{init}}$ are organized along the line axis $(\hat{\mathbf{v}}_k - \mathbf{x})$. The sampling range is calculated by finding the average height



(a) (b) (c)

Figure 18: Finding the optimal location using Geometric Constraints: (a) Samples ($\tilde{\mathbf{x}}_{ij}$) along lines from the evaluating point $\mathbf{x}$ and each projected VVPs $\hat{\mathbf{v}}_k$. Since the number of foreground samples is small, the window moves to top-right. (b) Evaluate Eq. 13 at each point $\mathbf{x}_{ij}$ inside $\mathbf{W}_{init}$. (c) Optimized location.



(a) (b) (c) (d) (e) (f)

Figure 19: Visualizing the components for evaluation of player location at time t and $t-a$. The player in the upper row remains in same location, and the one in bottom row moves. Colors range from blue (low) to red (high). In each row: (a) Values of $\mathbf{G}(\mathbf{x})$ in $\mathbf{W}_{init}$, (b) Values of $\mathbf{G}(\mathbf{x})^t$ in $\mathbf{W}_{opt}$ at current frame t, (c) Values of $\mathbf{G}(\mathbf{x})^{t-a}$ in $\mathbf{W}_{opt}$ at previous frame $t-a$, (d) Distances of the color histogram between frame t and $t-a$ within $\mathbf{W}_{opt}$, (e) Weighted sum of $\mathbf{G}(\mathbf{x})^{t-a}$ and $\mathbf{C}(\mathbf{x})^{t-a}$, and (f) A player of the evaluation and sampled colors. We extract velocity vectors between t and $t-a$ by subtraction of minima between (b) and (e). This vectors are shown in (b) as white arrow.

of players using vanishing points [31]. If the summation of all weights $\mathbf{PC}(\tilde{\mathbf{x}}_{i,k})$ for all views is too small (Fig. 18(a)), this is interpreted as a wrong initialization or a false-positive detection of the player and discarded.

To find the corresponding position $\mathbf{x}_{opt}^{t-a}$ of the player in the previous frame $t-a$

we establish a search window $\mathbf{W}_{opt}$ centered around $\mathbf{x}_{opt}$. We use a combination of the geometric constraints $\mathbf{G}(\mathbf{x})^{t-a}$ on the previous top-view frame $\mathbf{I}_{t-a}^{top}$ using Eq. 14, and a color proximity measure $\mathbf{C}(\mathbf{x})^{t-a}$:

$$\mathbf{x}_{opt}^{t-a} = \underset{\mathbf{x}(x,y)\in \mathrm{W}_{opt}}{\arg\min} \left(\mathbf{G}(\mathbf{x})^{t-a} + \beta \mathbf{C}(\mathbf{x})^{t-a}\right) \tag{14}$$

$\mathbf{C}(\mathbf{x})^{t-a}$ is a normalized Bhattacharyya distance of the color (HSV) histogram between the two sets of foreground samples used for $\mathbf{x}_{opt}^{t}$ and $\mathbf{x}^{t-a} \in \mathbf{W}_{opt}^{t-a}$ respectively (Fig. 19). The weighting factor β is usually very small (0.1). The use of color similarity reduces the chance that we are matching a different player. Once $\mathbf{x}_{opt}^{t-a}$ is found, we can define the motion (velocity vector) at $\mathbf{x}_{opt}$ as:

$$[u, v]^T = \frac{\partial \mathbf{x}(x,y)}{\partial t} \cong \left(\mathbf{x}_{opt}^{t} - \mathbf{x}_{opt}^{t-a}\right)/a \tag{15}$$

6.2.2 Dense Motion Field Construction using Spatio-temporal Radial Basis interpolation

Using our method for tracking player motion, we obtain a sparse set of motions on the ground plane. To generate a dense ground-level flow field we combine these sparse motions using radial basis function interpolation [19]. We also temporally interpolate the flow using a weighted set of motions over time.

As described in Section 6.2.1, the motion at a location $\mathbf{x}(x, y) \in \mathbf{I}^{top}$ is defined by a velocity vector $[\frac{\partial x}{\partial t} \ \frac{\partial y}{\partial t}]^T = [u \ v]^T$. If we detect N_k individual players at a given frame k, then the set of the positions is denoted as $\{\mathbf{x}_1^k, \mathbf{x}_2^k, \ldots, \mathbf{x}_{N_k}^k\}$, and the corresponding sets of velocities for each direction are denoted as $\{u_1^k, u_2^k, \ldots, u_{N_k}^k\}$, and $\{v_1^k, v_2^k, \ldots, v_{N_k}^k\}$ for x and y directions respectively.

We define a temporal kernel of size p, using a half Gaussian function. By applying the kernel to each entry of velocity over time, we can construct two $n \times 1$ vectors which are temporally smoothed versions of u_i^k to u_i^{k-p+1} and v_i^k to v_i^{k-p+1} respectively: $\mathbf{U} = [U_1, U_2, \ldots, U_{N_k}, \ldots, U_n]^T$ and $\mathbf{V} = [V_1, V_2, \ldots, V_{N_k}, \ldots, V_n]^T$, where U_i and V_i $(1 \leq i \leq n)$ are scalar velocities for each direction. The matching for each entry over

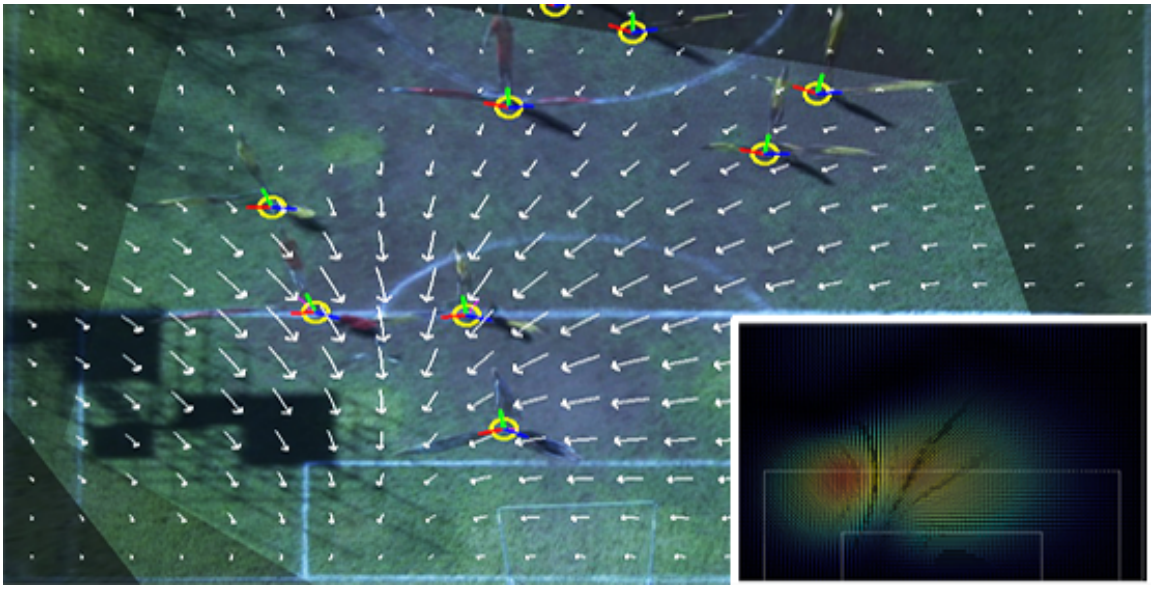


Figure 20: Motion Field Φ: White arrows represent the dense motion field generated from a sparse set of motions of players movements. Note that for visualization purposes the dense field is displayed sparsely by averaging the flow at each block.

time is set deterministically [135] (e.g. minimum distance and orientation). Note that commonly $n = N_k$ when the number of detected players does not vary over time. However, when there are less number of entries in a given frame k, compared to previous frames, n becomes larger than N_k.

Now, our problem may be stated as follows: given a collection of n scattered 2D points $\{\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}\}$ on the ground plane, with associated scalar velocity values $\{U_1, \ldots, U_n\}$ and $\{V_1, \ldots, V_n\}$, construct a smooth velocity field that matches each of these velocities at the given locations. Consider the scalar-valued functions $f(\mathbf{x})$ and $g(\mathbf{x})$ so that $f(\mathbf{x_i}) = U_i$, and $g(\mathbf{x_i}) = V_i$ respectively, for $1 \leq i \leq n$. For the case of interpolating velocity in the x-direction, we can express the interpolation function as:

$$f(\mathbf{x}) = c(\mathbf{x}) + \sum_{i=0}^{n} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|), \tag{16}$$

where $c(\mathbf{x})$ is a first order polynomial that accounts for the linear and constant portions of f, λ_i is a weight for each constraint, and $\mathbf{x}_i$ are the locations of the scattered points (*nodes*). Specifically, the radial function ϕ was chosen as the *thin-plate spline*,

$\phi(r) = r^2 \log r$, as it gives us C^1 continuity for smooth interpolation of the velocity field[1].

To solve for the set of weights λ_i so that the interpolation satisfies the constraints $f(\mathbf{x}_i) = U_i$, we solve the equation by evaluating each node at Eq. 16 (e.g. $U_i = c(\mathbf{x}_i) + \sum_{j=0}^{n} \lambda_j \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$).

Since the equation is linear in the unknowns, it can be formulated as a linear system:

$$\begin{bmatrix} \mathbf{A} & \mathbf{Q} \\ \mathbf{Q}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{U} \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 1 & x_1 & y_1 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix},$$

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_n]^T$, $\mathbf{c} = [c_1\ c_2\ c_3]^T$ and the $n \times n$ matrix $\mathbf{A} = (a_{ij}) = \phi(||\mathbf{x}_i - \mathbf{x}_j||)$.

Once we solve the system, the interpolated velocity of the x-direction at any location $\mathbf{x}_a(x_a, y_a) \in I^{top}$ can be evaluated as: $u_a = c_1 + c_2 x_a + c_3 y_a + \sum_{i=1}^{n} \lambda_i \phi(\|\mathbf{x}_a - \mathbf{x}_i\|)$. The velocity of y-direction is interpolated similarly. To generate temporally smoother transitions the flow is finally smoothed using a 1×5 box filter. We refer to this flow as the *motion field* on the ground, and denote it as $\Phi(\mathbf{x}) = f(\mathbf{x})\mathbf{i} + g(\mathbf{x})\mathbf{j} = u\mathbf{i} + v\mathbf{j}$. See Fig. 20.

6.2.3 Detecting Points of Convergence

The motion field reflects the local and global player motion representing the *play* (i.e. the strategy or intention of the players). We now define a *point of convergence* (POC) as the spatial location that play evolution is proceeding toward in the near future. In this section, we provide a method to detect POCs of the game by finding locations where the motion field merges.

Point of convergence detection is implemented in two steps. First, the motion field on the ground Φ, is used to propagate a confidence measure forward to calculate an

[1]Thin-plate spline minimizes the energy function : $E = \int_{\Re} (\frac{\partial^2 f}{\partial x^2})^2 + 2(\frac{\partial^2 f}{\partial x \partial y})^2 + (\frac{\partial^2 f}{\partial y^2})^2 dxdy$ over all interpolants [21, 131].

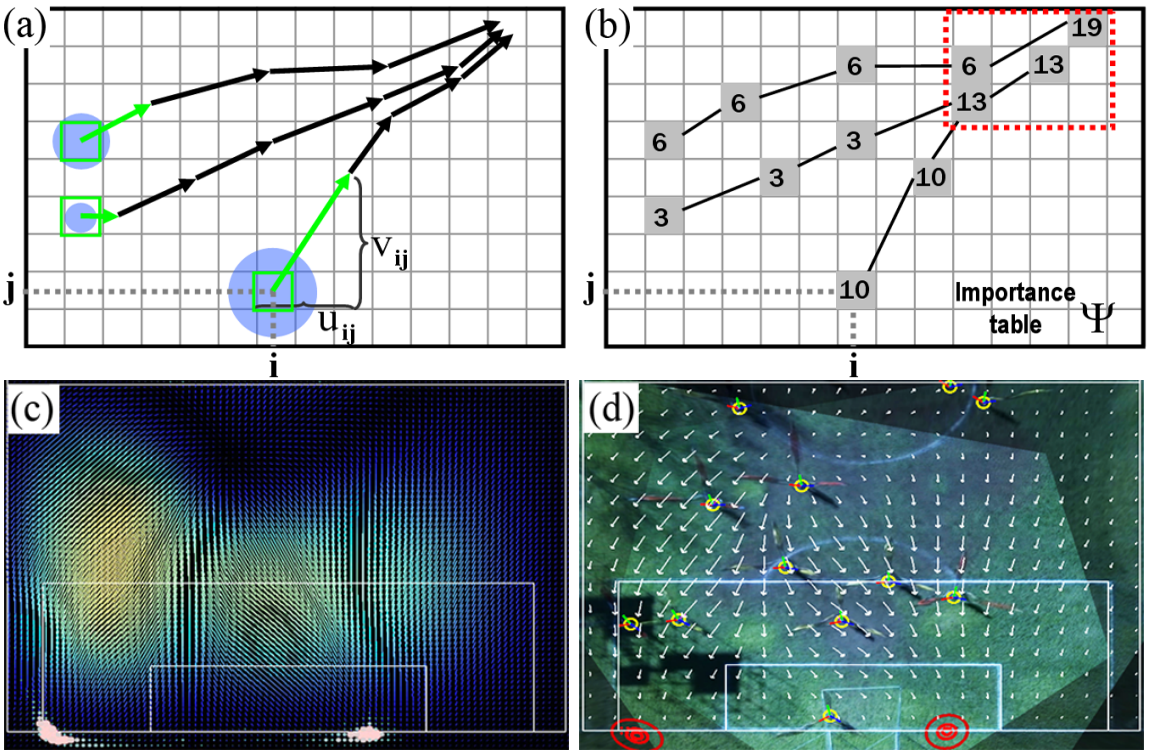


Figure 21: Points of Convergence detection: (a) Starting from the position $\mathbf{x}_{ij}$ and another point having a different magnitude of motion vector as an example. The magnitude of ρ_{ij} is propagated through the point $(i+u_{ij}, j+u_{ij})$. (b) An importance table Ψ is updated by adding propagated confidence along Φ. (c) Pink circles at bottom are the location where the accumulated importance is high enough (larger is higher confidence). (d) Meanshift clustering and Guassian mixture modeling detects two POCs in this case.

importance table Ψ whose size is the same as $\mathbf{I}^{top}$. Then, the accumulated confidence in Ψ are clustered and a Gaussian Mixture Model is used to detect POC clusters. Fig. 21 shows an example of how POCs can be automatically detected from a motion field Φ.

We introduce a confidence value, defined as the local magnitude of velocity at any location on the ground. In the first step, we propagate (copy) this value at a fixed time t from each starting location through Φ. Then, we accumulate these values along the trace in an *importance table* Ψ. Given a location $\mathbf{x}(i,j) \in \mathbf{I}_t^{top}$, Ψ is calculated by performing a forward propagation recursively based on the motion field Φ. The magnitude of the velocity $\rho_{ij}^2 = u_{ij}^2 + v_{ij}^2$ is propagated by updating Ψ as follows:

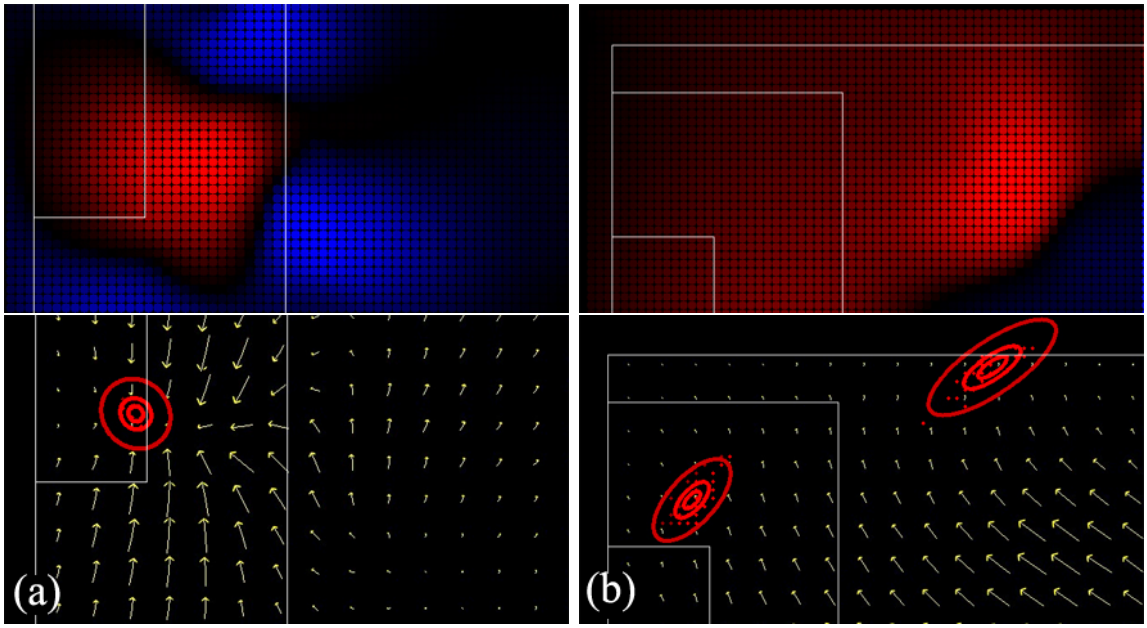


Figure 22: Divergence and Points of Convergence: Upper row: Red regions denote the regions where $\nabla\Phi < 0$, and blue regions denote regions where $\nabla\Phi > 0$. Lower row: Φ in (a) has specific singular sink, while Φ in (b) has no specific singular region. In both cases our approach detects the POCs (red elipses).

$\Psi(i + u_{ij}, j + v_{ij}) = \Psi(i + u_{ij}, j + v_{ij}) + \rho_{ij}$. We continue this forward propagation along the motion field until the attenuation which is proportional to $\rho_{i,j}$ is smaller than ϵ. Consequently, locations having a large ρ in Φ can have a large influence on far away locations as long as the motion field moves in that direction.

We compute the accumulated distribution of confidence Ψ (Fig. 21(c)), by computing confidence propagation for any location in $\mathbf{I}^{top}$. To determine the location and the number of POCs at a given frame k, we apply mean-shift clustering [26] to find an optimal number of clusters. Based on the initial mean and the number of clusters (modes), we fit a Gaussian Mixture model to the distribution of those regions using Expectation Maximization (EM) (Fig. 21(d)).

Note that our POC detection is different than classical singular (critical) points detection. Primarily, POC is a global measurement of the flow, while the critical point is a local extremum of the velocity potential and the stream functions [30].

The divergence of the motion field $\Phi(\mathbf{x})$ at the location $\mathbf{x} \in \mathbf{I}^{top}$ is defined as

$\text{div}(\Phi(\mathbf{x})) = \nabla\Phi = \frac{\partial \mathbf{u}}{\partial x} + \frac{\partial \mathbf{v}}{\partial y}$. If $\nabla\Phi$ is negative, the flux of the motion field across the boundary of the region is inward, while positive is outward. Thus, if the motion field flows to the boundary of a specific region, the divergence of that region becomes negative (see Fig. 22).

In practice, many of the detected POCs exist in regions where the local measurement of divergence becomes negative because the POC proceeds in the direction of the motion field flow. Therefore, in many cases, a POC exists where there is a significant singular sink point (Fig. 22(a)). However, if the majority of flows in a specific scene are not regular enough to construct an apparent local extremum, determining a POC by detection of singularities will fail (Fig. 22(b)). In such cases, our forward-propagation method can still locate regions of major global flows which signify positions where the play evolution may proceed to.

6.3 Results and Evaluation

We evaluate our approach with three different data sets from soccer, basketball and ice hockey matches. The soccer data set was collected by recording a match between a local college team and a local amateur team using three synchronized HD cameras. Each camera was mounted on top of an approximately $12m$ high scaffold. The basketball match data set was provided by the APIDIS project([9, 25]). In this data set, seven video cameras were used, including two wide angle views located at the top of the roof. Unlike the above two data sets, which were captured directly from real-life games, the third data set was captured from a simulated video game [37] using six virtual cameras. The AI agents used in the video game follow the behavioral model developed by the game company, and this model is apparently simpler than real human behavior. Thus, our data sets provides a good comparison of our approach on both human group behavior and an artificial group behavioral model.

Table 2: Detection errors on the number of cameras
False Positive (FP) and False Negative (FN) for each test. Two-views have larger FN error due to fewer constraints.

Views	FP (E1)	FN (E2)	Total
3	0.7162%	0.5649%	1.2811% (128/9913)
2	1.1000%	1.8120%	2.9120% (143/4901)

In this section, we first show qualitative and quantitative evaluation of extracting ground level motion on the soccer data set. Next, we show how our motion field interpolation is created and demonstrate POC detection using simple examples simulated by virtual agents having simple group behaviors. Finally, we evaluate the POC detection for the three data sets, and analyze various events from the different sports matches. We refer the reader to the accompanying video, which illustrates our results better than images.

6.3.1 Evaluation of Ground-level Motion Extraction

To evaluate the ground-level motion detection we back-project our automatic detection results onto each view and manually track them to find discrepancies (Fig. 23). We evaluated 14,814 individual players in 2,000 frames from one of our data sets. As we only track the players covered by at least two views, players seen in only one camera were not evaluated.

Table. 2 summarizes our results. Note that the false-negatives for two views are three times larger than using three views. This is due to the reduced number of constraints (vanishing directions) and fewer sample points that often results in a solution with a lower confidence rate. This tendency is also reported in other multi-view approaches ([71, 64]). Some qualitative evaluations for the ground-level motion extraction can be seen as the yellow circles in all result images in this section, but can be best viewed in the accompanying video.

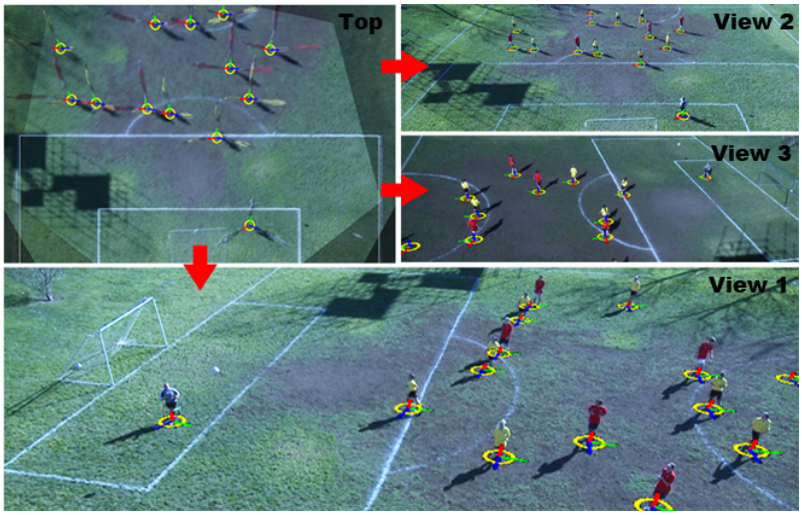
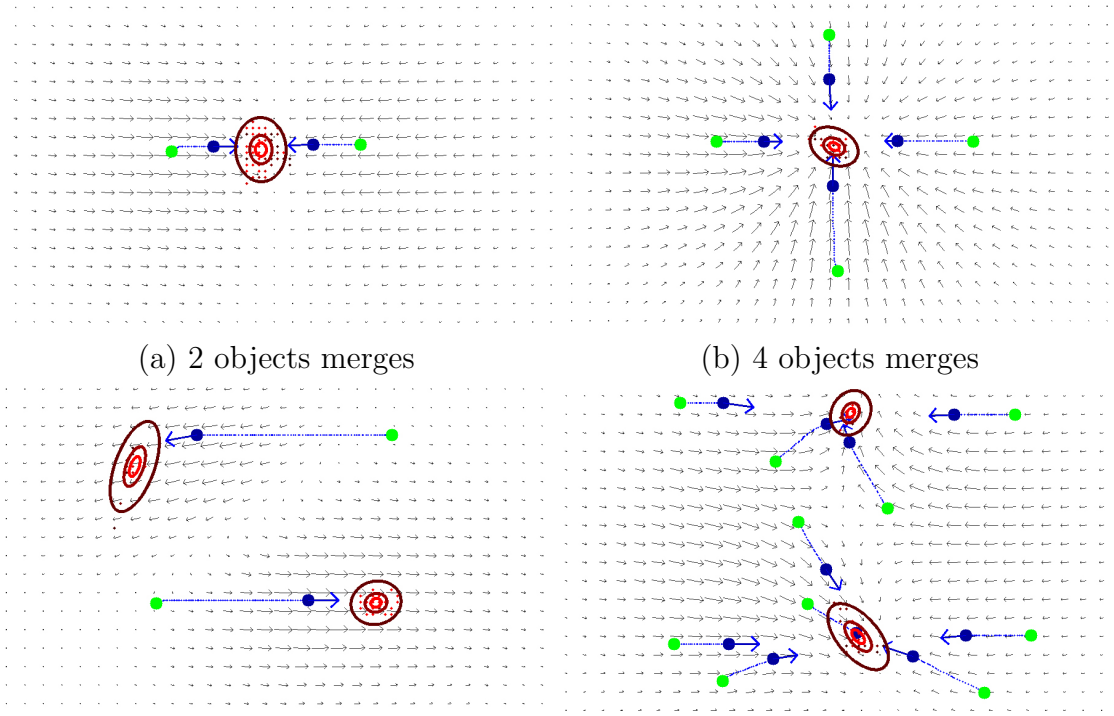


Figure 23: Evaluation of automatic player location on the ground: We back-project the position onto each view for evaluation. Each R, G and B line with yellow circle denotes the direction of the vertical vanishing points for each view (geometric constraints).

6.3.2 Evaluation of Motion Field and POC detection

We first evaluate how well a flow field generated from a sparse set of ground level motions can represent play evolution, and show that POCs detected from the flow field are reasonable. Fig. 24 illustrates motion field generation and POC on a set of simulated agents. We used a simple rule-based behavioral model ([111, 112]). This simulation allows us to manipulate the situations that reflect a variety of events that are likely to happen in dynamic sports scenes. For simple merging movement of players, as depicted in Fig. 24(a) and (b), the flow field generates a critical point, and our algorithm can detect it well. However, also when two players are heading in opposite directions (Fig. 24(c)) and in more complex movements (Fig. 24(d)) we also manage to find the POC as discussed in Section 6.2.3, and Fig. 22.



(a) 2 objects merges (b) 4 objects merges

(c) 2 objects crosses (d) complex example

Figure 24: Flow generated from simple motions, and their POCs: Blue circles are the moving objects, and the arrows indicate current velocity. Green circles are the location where each object started from. Blue dots are the trace of each object's past location. Black arrows indicate the block-average of the flow field. Red contours represent the calculated POCs.

6.3.2.1 Qualitative Evaluations

Evaluation of the motion field generation and POC detection for a variety of our sports game datasets are demonstrated from Figs 25 to 32 (See the descriptive captions for each figure). In all figures, the distributions of POC clusters are shown as red iso-contours around the POC, and the motion field on the ground is depicted with white arrows. These figures, and others in the accompanying video, illustrate how our motion fields and POC detection can reveal interesting game events and foresee important future positions on the field.

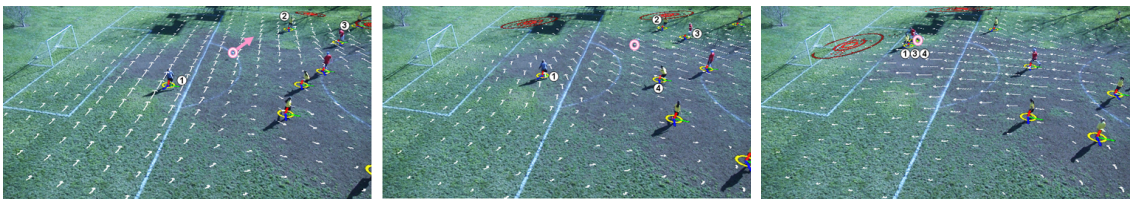


Figure 25: Example 1 of play evolution. In all figures, the pink circles are the location of the ball, the arrows denote the initial direction of the movement of the ball, white arrows denote the motion field on the ground, and red contours are the distribution of POC where play evolution aims. In this example - interception & goal keeping. **(Left)** Goalkeeper 1 passes a ball to the last defender 2. A POC appears near the defender, **(Middle)** While the ball is still on the way to the defender 2, another POC appears at different location as offender 3 approaches and the goalkeeper and defender 4 decide to defend. **(Right)** The ball is intercepted by offender 3 and the POC event in the left region actually occurs. Finally, the ball was saved by the goalkeeper. Note that two POCs appear in both possible directions.

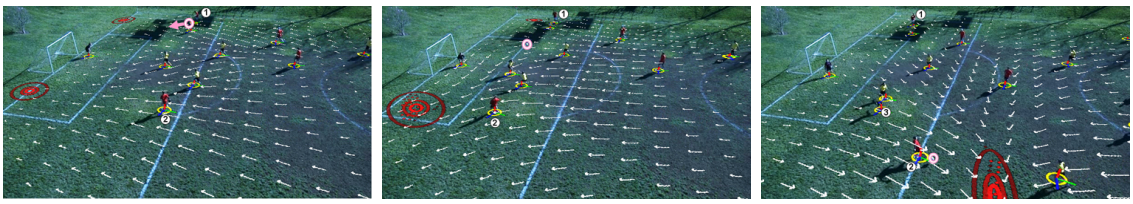


Figure 26: Example 2 of play evolution - center pass (Left) Offender 1 dribbles towards the upper corner while offender 2 runs toward the other corner (two POCs). **(Middle)** Offender 1 kicks a center pass. While the ball is travelling the POC near offender 2 becomes larger. **(Right)** Defender 3 intercepts and the ball changes its direction. Offender 2 and another defender approach to the ball.

6.3.2.2 Quantitative Evaluation

Verifying that our approach can *predict* important future positions is not a simple task since, (1) the position of important regions in the game can be subjective, and (2) there is no ground truth for defining these regions. To give some quantitative measure, and since ball location is one of the important positions in any ball game, we evaluate the results of trying to predict the future location of the ball using our POCs. We assume that the current POC will be a good indication to where the ball will be in some point in the future, and then compare the two by collecting ground truth of the ball position manually. To investigate how well and how far in advance

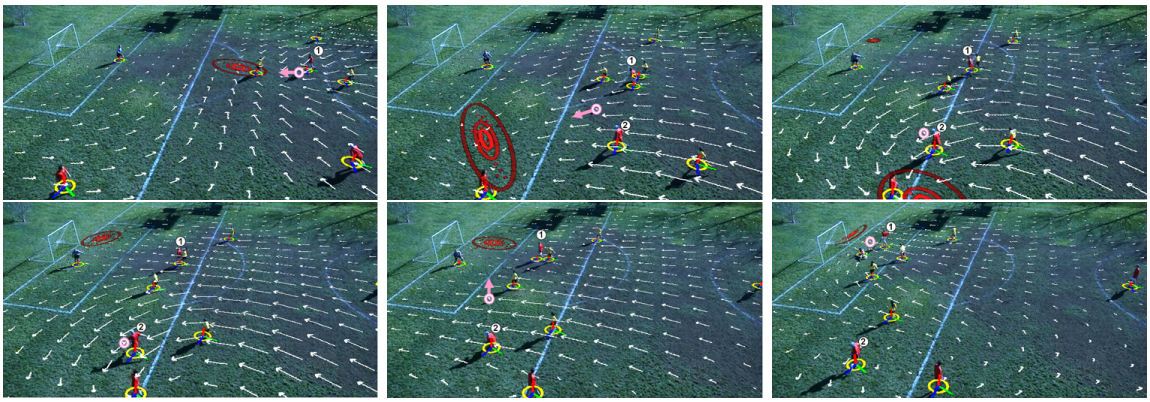


Figure 27: Example 3 of play evolution - back-door and through pass: **Upper row: (Left)** Offender 1 dribbles forward (a POC is in front of the player). **(Middle)** Offender 1 passes the ball to offender 2 as part of a back-door strategy. A POC appears near offender 2. **(Right)** While the offender 2 waits to receive the ball, offender 1 and the defenders are running toward the goal-post. Another POC appears near goal. **Lower row: (Left)** Before offender 2 kicks the ball, the POC near the goalpost becomes larger. **(Middle)** Offender 2 through-passes to offender 1. **(Right)** Offender 1 attempts to score.

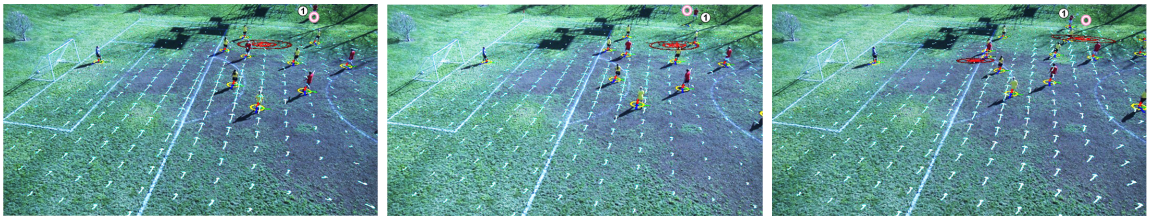


Figure 28: Example 4 of play evolution - Outbound and Throw-in: **(Left)** Ball was out-bounded, and Offender 1 has the ball. **(Middle)** Offender 1 is looking for the location to trow-in. **(Right)** Offender 1 throws the ball in.

Figure 29: Example 5 of play evolution in Basketball - Turn over: **(Left)** Yellow team just got the score at the right half-court (outside of the scene). **(Middle)** Most of the offenders and defenders are getting back to the left half-court. **(Right)** Offender 1 brought the ball into the left half-court (Turn over).

our estimated POCs reflect the future ball location, we vary the temporal offset for measuring the ball location from 4 frames to 120 frames from the current frame.

Figure 30: Example 6 of play evolution in Basketball - Pass and Three point shot: **Upper row: (Left)** Offender 1 just received the ball. a PoC appears at the location of the player, as the defender 3 is approaching to defend against the offender 1. **(Middle)** Offender 1 passes the ball to another offender 2 (Shooting Guard), and the defender 3 moves to the location of the offender 2 (another PoC appears at the offender 2). **(Right)** Offender 1 moves inside to receive the ball again (The first PoC moves to the location). **Lower row: (Left)** Offender 2 directly shoots rather than passes the ball to the offender 1. Defender 3 try to block the shot. **(Middle)** Because the ball is approaching to the basket, the new PoC appears in the region near under the basket. **(Right)** Finally the ball enters the basket and the PoC moves to the location under the basket.

First we show how the distances over different temporal offset reflects the game events. Fig. 33 (upper) shows the distance between the current POC and the ground truth future position of the ball (determined manually in the video) using different offsets. We observe that sometimes the distance is smaller for the near future (blue line) than for distant future (red), while at other times this trend is reversed. This observation motivates us to differentiate the two cases, and we will be referring to the former as "*short-term prediction*" (small prediction error in near future) and to the latter as "*long-term prediction*" (small prediction error in distant future). Therefore, the short-term prediction indicates that players in the field rarely anticipate longer terms, while the long-term prediction indicates that they can decide their reaction for farther events.

To investigate when short-term or long-term prediction occurs, we track additional

Figure 31: Example 7 of play evolution in Basketball - Fake and Drive-in shot: **Upper row: (Left)**Offender 1 has the ball, and the defender 2 covers him. **(Middle)** Offender 1 dribbles past the defender. **(Right)** Defender 3 tries to cover offender 1. Other offenders at left and right side of offender 1 are ready to receive the ball. **Upper row: (Left)** Offender 1 pivots to drive by faking defender 3. **(Middle)** Offender 1 drives past defender 3, and PoC near basket appears. **(Right)** Finally the offender 1 successfully drives in.

Figure 32: Example 8 of play evolution in Ice Hockey - Stickhandling and shoot: **(Left)** Offender 1 (white team) stick-handles to the goalcage, and PoC appears near the player. **(Middle)** Offender 1 shoots, and the puck is about to pass by the goaltender. Additional PoC near the left side of the goal cage indicates the shot will not be successful. **(Right)** The puck is passed away and bounced to the upper side of rink. Some players are heading to the puck.

information from the video. The plots in Fig. 33 (lower) show the height, velocity and derivative of the angle of direction of the ball. Additionally, we add high-level game states such as "in-bound" (ball is inside the game field) and "out-bound" (ball is outside the field) as horizontal color bars in Fig. 33 (lower). Based on this information, we observe that, (1) short-term prediction happens during in-bound states, (2) it is more likely to have long-term prediction during out-bound, and (3) during in-bound

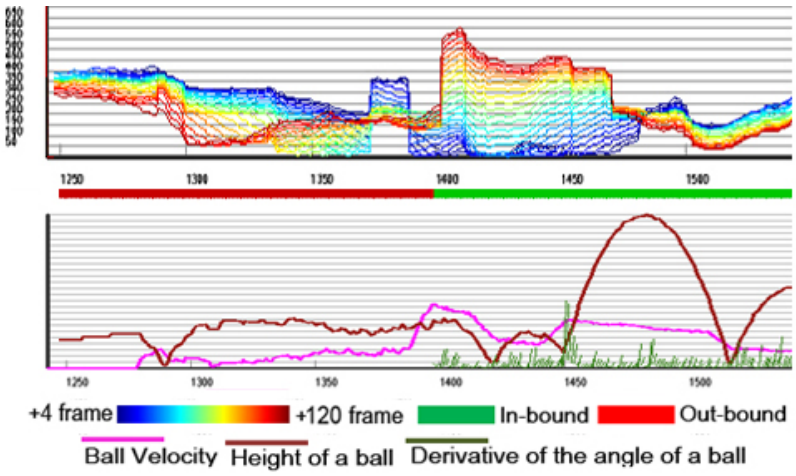


Figure 33: Example of the quantitative evaluation for our approach: **(Upper)** *Distances between the location of the detected* POC *in the current frame and the location of the ball in future frames.* The measurement varies from blue (4 frames later) through red (120 frames later). For both graphs, x-axis is the frame number, and y-axis is a pixel level distance (100 pixels is approximately 4 meters). **(Lower)** Ball Information, and game status (red : inbound, green : outbound)

states, if the ball position is high and speed of the ball is decreasing, long-term prediction is more likely.

These data-derived observations appear physically plausible. For example, during in-bound states, the game speed is faster, and players do not have time to estimate longer durations. Out-bound states usually the case when the game is suspended and turning over. Thus, players can easily predict where the game is going to proceed in long-term manner. If the ball position is high in the air, players have more time to evaluate where the ball will fall in the long-term. However, as we will show in later part of this section, game contexts can also affect these observations when some of players can expect something based on strategic transition.

Based upon the observations and the method of quantitative evaluation that we drew above, we analyze each of specific events shown in Figs 25 to 32. Each graph in Figs 34 to 41 represents the distance between the location of detected POC and the location of the ball location in future frames, and is analyzed as follows:

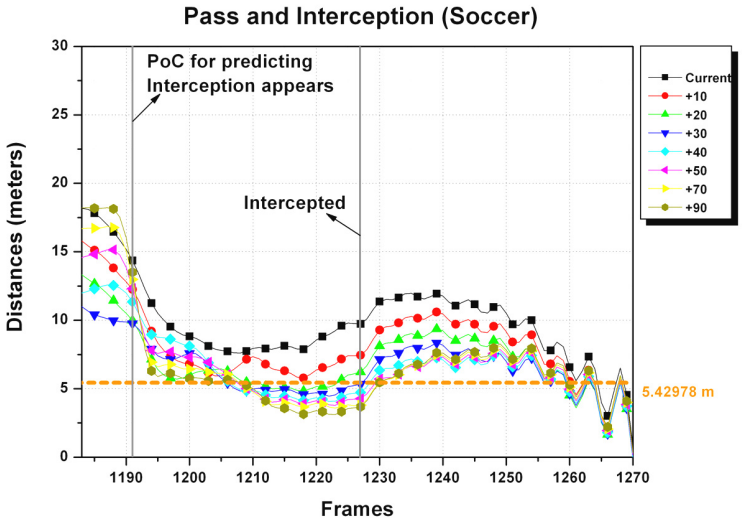


Figure 34: Pass and Interception: The distance between current POC and the ball locations varying from current frame (Black plots) to 90 frames later (dark yellow plots). See the event depicted in Fig. 25 and Fig. 15

Soccer: Pass and Interception (Fig. 25) In this example an offender is trying to intercept the ball. Some of the defenders including the goal keeper are able to foresee the interception. Because of the movement of offender 3 as he approaches to catch the ball. Hence, we expect that the distance graph will represent long-term prediction. Indeed, the graph in Fig. 34 shows that the prediction distances in near frames (black, red and green representing current to 20 frames offsets) is larger than those in far frames.

Soccer: Center Pass (Fig. 26) The center pass (an offender passes a ball to the other offender in the center-field or in the other corner) often happens during soccer play, and it is predictable that players will head in separate directions and some players will be looking for the best location to receive the ball. Thus, we can expect that before the center pass happens, it would be more likely for the POC to represent long-term prediction, until the game returns to play-by-play again (more likely to represent short-term). In the graph in Fig. 35, we can see long-term prediction appears first (black to yellow), then it moves to short-term (yellow to black). However, as

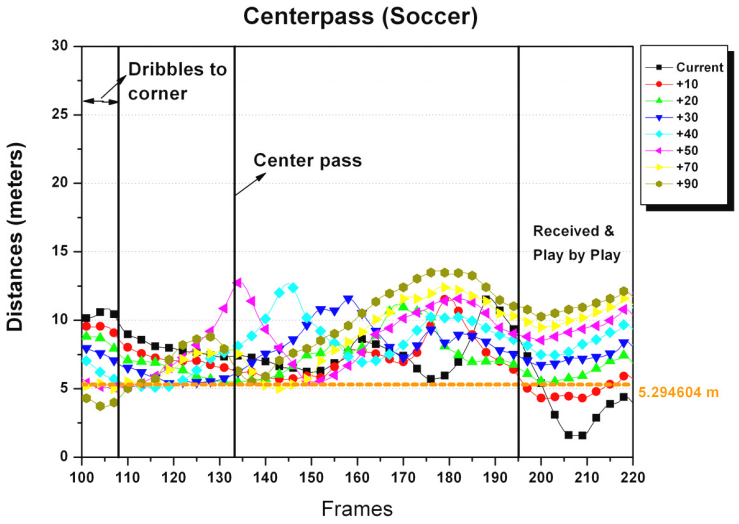


Figure 35: Center Pass: See the event in Fig. 26

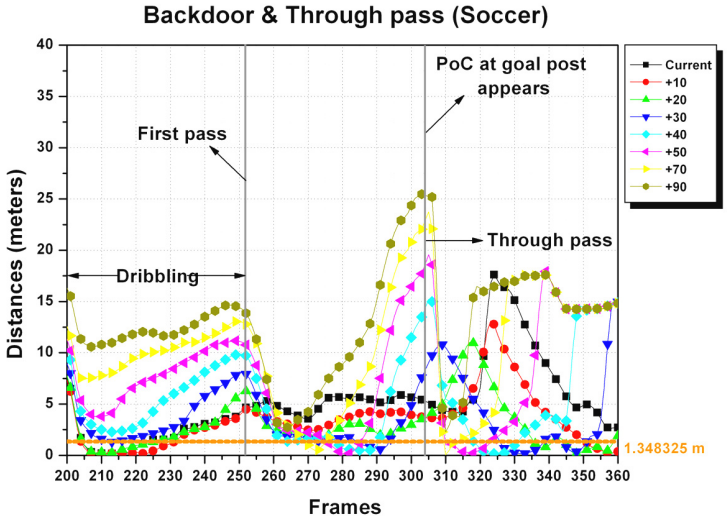


Figure 36: Through pass: See the event in Fig. 27

shown in the event (Fig. 26), the pass was blocked by a defender, and this results in irregular transitions between longer term and short term: the blue (40 frames offset) to dark yellow (90 frames offset) do not show smooth transitions.

Soccer: Back-door and Through Pass (Fig. 27) Back-door play followed by the through pass is one of the most popular strategies in many team sports games. Thus, both the offender who is supposed to receive the ball and the defender who covers him

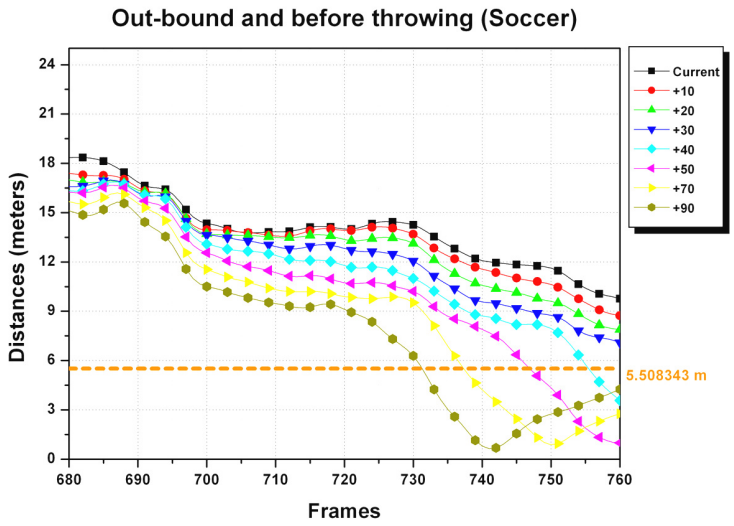


Figure 37: Out-bound and Throw-in: See the event in Fig. 28

always have to be ready and predict where the other offender will run to. The graph in Fig. 36 first shows a play by play tendency with short-term predictions when the player dribbles. After the first pass, most of the plots follow long-term predictions except the cyan to dark yellow plots (50 to 90 frames offset). This result can be interpreted as the maximum offset for this prediction, and is at most 50 frames in this event. Note in the video that 50 frames after the first pass is the time when the POC starts moving towards the goal area. This effect can be seen in the graph: the frame difference between first pass and through pass is almost 50 frames.

Soccer: Out-bound and Throw-in (Fig. 28) As we already discussed earlier, the out-bound play is more likely to be presented by long-term predictions. This is because, especially in a soccer game, every player can predict approximately where the ball will be thrown to. Thus, they can easily prepare to find a better position in the field. The graph in Fig. 37 illustrates how such event always represents long-term prediction.

Basketball: Turning over (Fig. 29) Similar to the out-bound play in a soccer game, a turn over in basketball is characterized by long-term prediction for players. Once a turn over happens, the offending team is switched to a defending team, and

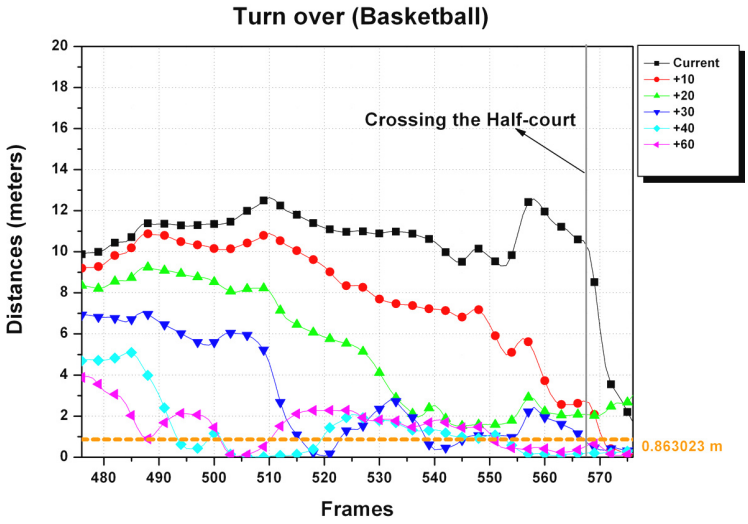


Figure 38: Turn over: See the event in Fig. 29

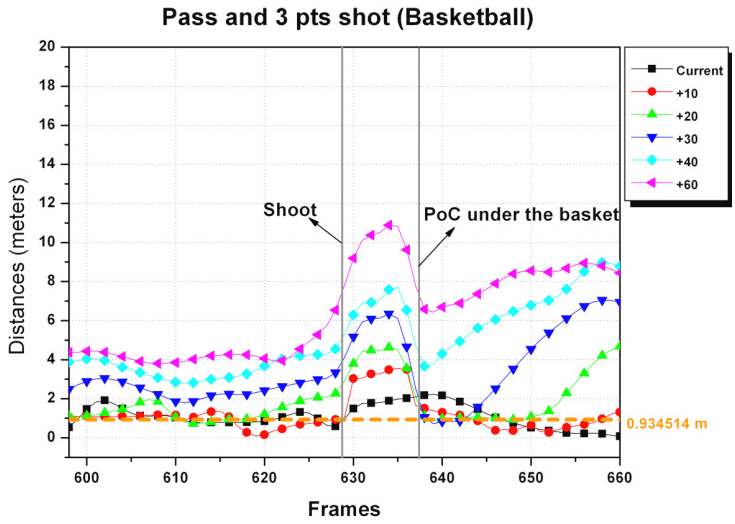


Figure 39: Three point shot: See the event in Fig. 30

most of the players move back to the other half of the court. The examples in Fig. 28 shows such an event, and the graph in Fig. 38 shows that a turn over usually follows a long-term prediction.

Basketball: Three Point Shot (Fig. 30) In this event, the POC is well located when offender 1 receives the ball. After that, the game changes to play-by-play so that the actions are immediately performed. When offender 2 shoots for three points,

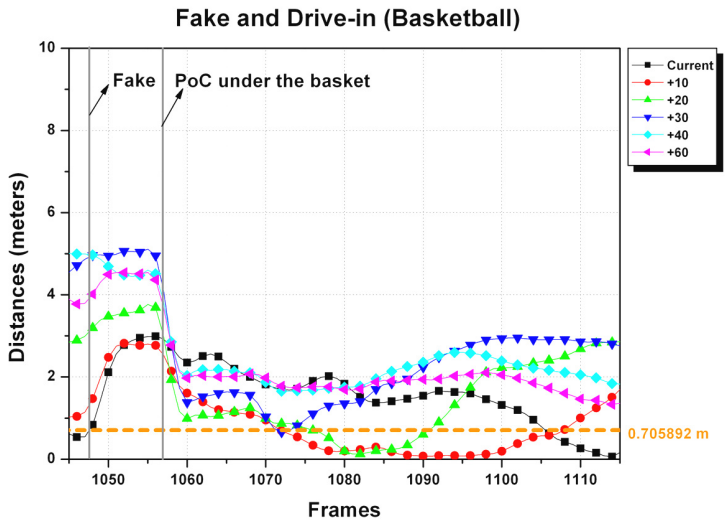


Figure 40: Fake and drive-in: See the event in Fig. 31

both offenders and defenders suddenly react to prepare for the rebound. Thus, we can easily expect that next POC will be positioned under the basket, and this prediction can be seen as a long-term. As expected, the graph in the Fig. 39 represents a short-term prediction before the shot is taken, and when the algorithm detects a POC, it presents a long-term prediction (frame 636 to 645). However, the graph shows that the maximum offset during the long-term prediction is approximately 30 frames (blue plots). This is reasonable because the ball enters the basket at the frame 660.

Basketball: Fake Spin and drive-in (Fig. 31) As the offender tries to leave behind several defenders before he drives in, the distance tendency is short-term (Fig. 40). The interesting observation here is that because the defenders and the other offenders were faked by the offender who will make a drive-in with fake spin, their reactions directly result in a wrong prediction as *faked* POCs appear on both side. This situation is depicted in Fig. 31(top, right), and the graph shown at Fig. 40 indicates this situation as some of the distances are suddenly increased near frame 1050. Then, the tendency moves to long-term for a while as soon as the players recover from their error, as shown at end of the graph.

Ice Hockey: Stick Handling and Shot (Fig. 32) As already noted, this data set

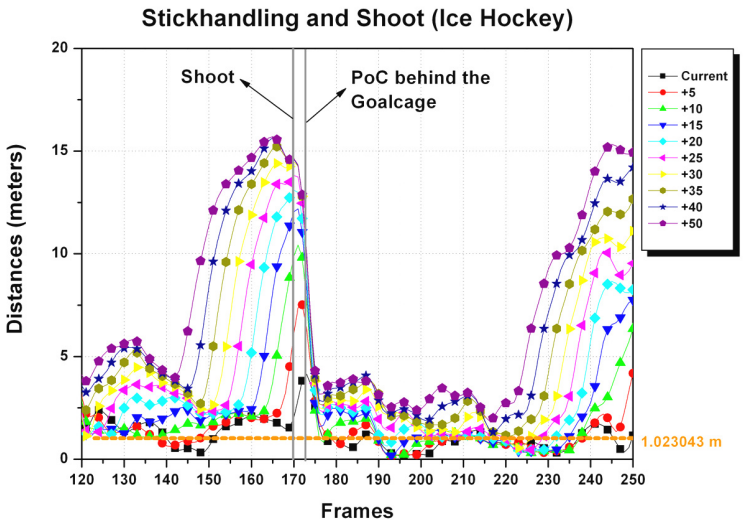


Figure 41: Stick Handling and Shot: See the event in Fig. 32

was captured from a simulated video game [37]. Thus, we can assume that (1) the behaviors of players are not as complex (or structured) as true human players due to a simpler behavioral model, and (2) the speed of play-by-play is faster because ice hockey is a fast game, and we expect the predictions are more likely to be short-term. The graph in the Fig. 41 indeed shows a short-term prediction tendency, as expected, and demonstrates that our approach also works reasonably well even for artificially generated behaviors.

So far, we have analyzed each event based on the distance between predicted POC to the ball locations in different time off-sets to see *how* our method can foresee the future ball locations. To verify how close our method can locate the position of the future ball location regardless of the game event and the temporal offsets, we measure the average of the minimum distance found over all time offsets. Each of the average of minima is noted as orange horizontal line at each graph, and these results are summarized in Table. 3. For the Soccer game the average error is 3.0% of the field area. The basketball data has an error of around 0.56%, and the ice hockey data has 0.2%.

Table 3: Average errors of minimum distances
Average errors of distances on various time offsets from examples shown in Fig. 25–32. The graphs representing detailed analysis for each example are described in Figs. 34 to 41.

Sports	Event	field size	avg' min'	Error
Soccer	Interception	62×37 m^2	5.43 m	4.0 %
Soccer	Center pass	62×37 m^2	5.29 m	3.8 %
Soccer	Through pass	62×37 m^2	1.35 m	0.25 %
Soccer	Throw-in	62×37 m^2	5.51 m	4.15 %
Basketball	Turn over	26×15 m^2	0.86 m	0.59 %
Basketball	3 pts shot	26×15 m^2	0.93 m	0.69 %
Basketball	Drive in	26×15 m^2	0.71 m	0.4 %
Ice hockey	S.H.Shot	61×26 m^2	1.02 m	0.2 %

Table 4: Computational time with various sizes of top-view
Test results using an Intel Core i7-965, 3.2GHz with 3GB RAM.

Field Resolution	Motion	Motion Fields	POC
670×500	166.33 ms	15.5 ms	65.10 ms
945×700	325.5 ms	36.5 ms	254.4 ms
1350×990	871.2 ms	93.5 ms	568.24 ms

The computational time for each part of our approach is given in Table 4. Note that the computation is highly dependent on the resolution of the field ($\mathbf{I}^{top}$). Our current implementation is not real-time, but it could be optimized.

6.4 Summary

We introduce a novel approach for play evolution analysis using multiple views. We detect and track the ground-level motion of players through optimization of geometric constraints. Using these sparse sets of tracks, we generate a dense motion field on the ground-plane and detect points of convergence in the field as possible future interesting locations of play evolution. We evaluate our approach both quantitatively and qualitatively using data sets from variety of team sports game showing that ours can robustly predict the location where the play will evolve.

In the future, we plan to develop more efficient ways to extract the motion fields and reduce the computational cost. In addition, we will investigate how the transition of detected POC can reveal the game context as we demonstrated in the evaluation section. To this end, we are interested in pursuing robotically controlled cameras based on the scene and play evolution analysis to realize our goal of automated broadcasting.

CHAPTER VII

MOTION PATTERN RECOGNITION AND ANOMALY DETECTION FROM SPATIO-TEMPORALLY SPARSE MOTION TRAJECTORIES

As described in Chapter 4, representations of motion patterns can also be generated by scattered data approximation. In this chapter, we show the use of such representations generated from a stochastic process for recognizing motion patterns and detecting anomalous events. We also show that the proposed motion representation is an effective solution for motion recognition tasks, even with various data constraints that we may often face in real-world situations. Such constraints of the input video include (1) a varying frame rate resulting from various exposures, (2) the sparse frame rate from either time-lapse style video or large-scale aerial videos, and (3) incomplete online motion trajectories at a specific moment.

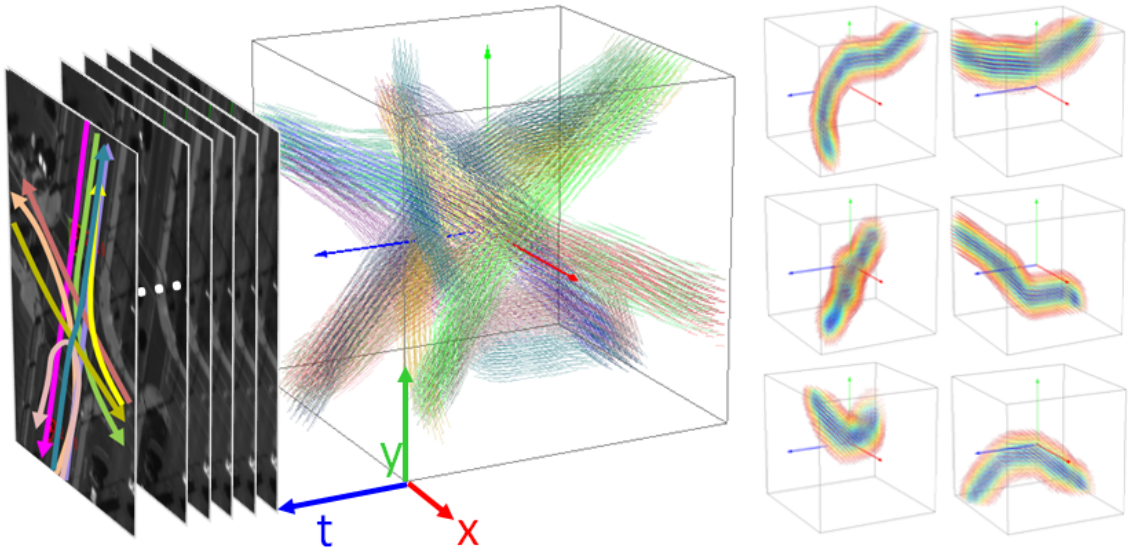
7.1 Stochastic Motion Representation to Motion Pattern Recognition

Motion trajectories of moving targets are essential for providing information about an object's movement patterns over time. Visual analysis of such moving objects in videos, is therefore a considerably important topic [54, 86, 102, 137, 147]. A key element in such analysis is representing the trajectory patterns of objects in a manner that allows for effective discrimination of these trajectories from specifically known and unknown patterns. Such forms of discrimination are usually evaluated by comparing spatial proximity between trajectory points with known (or learned) reference patterns or by clustering all the input trajectories to find similar patterns [54, 91, 151].

Such approaches for analysis of trajectories are useful for recognizing motion patterns that are representative of the data, but do not necessarily support (1) detecting complex time-varying patterns, caused by subtle changes of acceleration or sudden stops, which may be important for predicting changes in headings or detecting unknown or anomalous patterns, or (2) recognizing incomplete trajectories for instant and online monitoring. Such instances require modeling and analyzing motion trajectories in a spatio-temporal domain. Furthermore, constructing reference models from videos having (a) sufficiently *representative* patterns for sequences of varying lengths, and (b) different *temporal sampling* (frame-rates) remains a difficult task. The goal of the research in this chapter is to present a representation that allows for matching of complex trajectories. We demonstrate the validity of this representation for observing and modeling traffic patterns from video.

Specifically, in this chapter, we propose a novel representation of motion trajectories using Gaussian Process Regression (GPR). This representation supports effective recognition of normal and anomalous patterns by generating a continuous vector field, a Gaussian Process Regression Flow (GPRF) (Fig. 42). We demonstrate using this framework (Fig. 43) a process to take visual observations (trajectories) and generating a flow field that provides *representative motion tendencies* even with a minimal amount of data in each training set. Furthermore, this representation is invariant to temporal sampling constraints (*i.e.*, frame-rates of data can vary from trajectory to trajectory). Our use of flow fields, which describe statistical certainties in terms of their spatial shape and temporal tendency, can effectively classify and predict motion patterns, and detect anomalous tracks from incremental online trajectories. We show that the proposed spatio-temporal flow can efficiently model complex motions, such as stopping or accelerating of cars in traffic videos.

The main contributions of the study in this chapter are: **(1)** A novel method for modeling a trajectory as a 3D continuous dense flow field from sparse set of



(a) (b)

Figure 42: Gaussian Process Regression Flows (GPRF): **(a)** Video frames having trajectories, and the normalized *mean flows* representing 17 different *spatio-temporal patterns* of motion trajectories are visualized in different colors. Each mean flow is generated from trajectories extracted from videos using Gaussian process regression. We classify each online track by traversing inside the flow fields and collecting posterior densities along their paths. **(b)** Some of individual mean flows are shown : blue indicates that the mean flows in the region have lower variances (and thus higher certainty), and red indicates higher variances.

Figure 43: Overview of our framework: From tracks to GPRF.

vector sequences using Gaussian Process Regression; this enables modeling complex patterns, especially as seen in videos of traffic. **(2)** A random sampling strategy for learning stable multiple classes of flow field; this generates a fair distribution of certainties and alleviates unwanted variation. **(3)** A set of effective metrics to compare an input trajectory (incomplete or complete) to learned flow fields. And, **(4)** a locally dominant ratio for effectively discriminating a correct pattern from incorrect ones, and calculating ambiguities used for detecting anomalous patterns.

7.2 Gaussian Process Regression for Flow

To model time-varying motions, we first extend the motion trajectories extracted from video sequences into the spatio-temporal domain. We define a trajectory as a discrete vector sequence in $\Re^3$. Let $x \in \Re^3$ be the position (u, v, t), and the position sequence be $\mathbf{x} = \{x_1, \ldots, x_n\}$. Moreover, let $y \in \Re^3$ be the velocity of each direction (y_u, y_v, y_t) [1], and the velocity sequence be $\mathbf{y} = \{(y_{1,u}, y_{1,v}, y_{1,t}), \ldots, (y_{n,u}, y_{n,v}, y_{n,t})\}$. We also denote the sequence of each velocity component as: $\mathbf{y}_u = \{y_{1,u}, \ldots, y_{n,u}\}$, $\mathbf{y}_v = \{y_{1,v}, \ldots, y_{n,v}\}$, $\mathbf{y}_t = \{y_{1,t}, \ldots, y_{n,t}\}$. The trajectory sequence of n vectors are defined as $\mathrm{T}_n = \{\mathbf{x}, \mathbf{y}\}$. We first briefly introduce Gaussian Process Regression (GPR) and explain how we use it for modeling the dense vector field from the set of sparse vector sequences (Fig. 43).

7.2.1 Constructing Posterior Density

We consider the regression model $y = f(x) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. We use a Gaussian process model on f with a mean function $m(x) = \mathbb{E}[f(x)] = 0$ and a covariance function $K(x, x'') = \mathbb{E}[(f(x) - m(x))(f(x'') - m(x''))] = \mathbb{E}[f(x)f(x'')]$. The observation vector $\mathbf{y} = \{y_1, \cdots, y_n\}$ then follows a zero-mean multivariate Gaussian with a covariance matrix, $\mathbf{K}^* = \mathbf{K} + \sigma^2\mathbf{I}$, where $[\mathbf{K}]_{ij} = K(x_i, x_j)$. The posterior density for a test point x^*, $p(y^*|x^*, \mathbf{x}, \mathbf{y})$, is a univariate normal distribution with the mean $\bar{y}^*$ and the variance $\mathrm{var}(y^*)$:

$$\begin{aligned} \bar{y}^* &= k(x^*)^T (K^*)^{-1} y \\ \mathrm{var}(y^*) &= K(x^*, x^*) - k(x^*)^T (K^*)^{-1} k(x^*), \end{aligned} \tag{17}$$

where $k(x^*) = [K(x^*, x_1), \ldots, K(x^*, x_n)]^T$. In our case, we train a separate GP regression for each of $\mathbf{y}_u$, $\mathbf{y}_v$, and $\mathbf{y}_t$ at a given point set $\mathbf{x}$.

We chose Gaussian Automatic Relevance Determination (ARD) kernel [27] as

[1] y_t is a frame difference, thus it is usually a constant. See Section 7.2.2

the covariance function. We select its hyper-parameters using the limited memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimizer [95] by maximizing the marginal log-likelihood.

7.2.2 Mean Flow and Confidence Band

Since we want to model a continuous flow field where the function represents velocities for each spatial direction and the uniform temporal grid (frame difference) at any point $x = (u, v, t)$, we calculate a GP regression model for each velocity component. Hence, each of the posterior densities $p(y_u^*|x^*, \mathrm{T}_n)$, $p(y_v^*|x^*, \mathrm{T}_n)$, and $p(y_t^*|x^*, \mathrm{T}_n)$ are computed separately. We can then express the mean flow (see Fig. 44) as a vector field:

$$\Phi(x) = \bar{y}_u^*(x)\mathbf{i} + \bar{y}_v^*(x)\mathbf{j} + \bar{y}_t^*(x)\mathbf{k} \in \Re^3, \tag{18}$$

with a variance for each velocity component $\mathrm{var}(y_u^*(x)), \mathrm{var}(y_v^*(x)), \mathrm{var}(y_t^*(x))$ respectively. The sequence of mean flow vectors with minimum variances over each time u–v slice (see Fig. 44) defines the approximated version of input trajectories.

We denote the pair of the mean flow Φ and its variances as *Gaussian Process Regression Flow (GPRF)*, the size of output variance in posterior density having 95% of certainty ($1.96\sigma^2$) as a *confidence band (CB)*, and the sequence of mean flow having a minimum variance over each time grid as a *Approximation of Learned Trajectory (ALT)*.

7.3 Learning Stable GPRFs

In learning GPRFs, there are two issues we need to address: (1) How to model a GPRF from different trajectories, which may have different lengths, and (2) How to handle multiple GPRF models trained from different numbers of trajectories with heterogeneous scales and frame rates. Before applying the Gaussian process regression

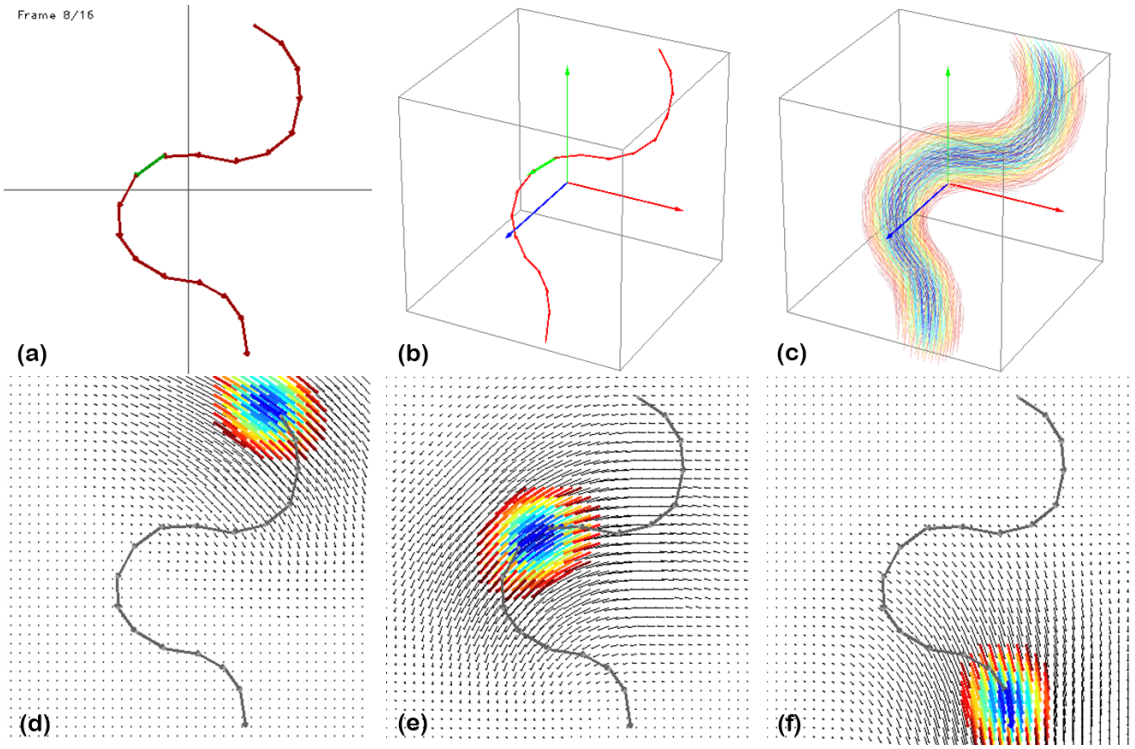


Figure 44: Example of GPRF, and mean flows: **(Upper row)** **(a)** Sample trajectory having a duration of 16 frames: each arrow starts from the location of $x(u, v, t)$, and has the velocity values **(b)** The trajectory in (u, v, t) space $\in \Re^3$ **(c)** GPRF generated from the trajectory **(Bottom row)** The projected mean flows in each of $u - v$ slice at $t = 0$ **(d)**, $t = 8$ **(e)**, and $t = 16$ **(f)** respectively. Mean flows with different levels of confidence exist in any grid points in the space. In each image, only mean flows having the variance of less than half the maximum variance among 95% confidence are shown as color values. The colors vary from the larger posterior variances (red) to smaller variances (blue).

framework, we need to normalize the length of the tracks used for constructing classes (Section 7.3.1).

7.3.1 Normalization

Unlike alignment-based approaches (i.e. DTW and LCS) [104, 137], our flow field uses a normalized *frame*, which is similar to the method introduced in [86]. Therefore, the time axis is discretized into l equal sized frames. In practice, when we construct multiple GPRFs from different training sets, the positions of each sample along the t axis and their velocity components of $\mathbf{y}_t$ should be normalized by $\frac{l}{n_i}$, where n_i is the

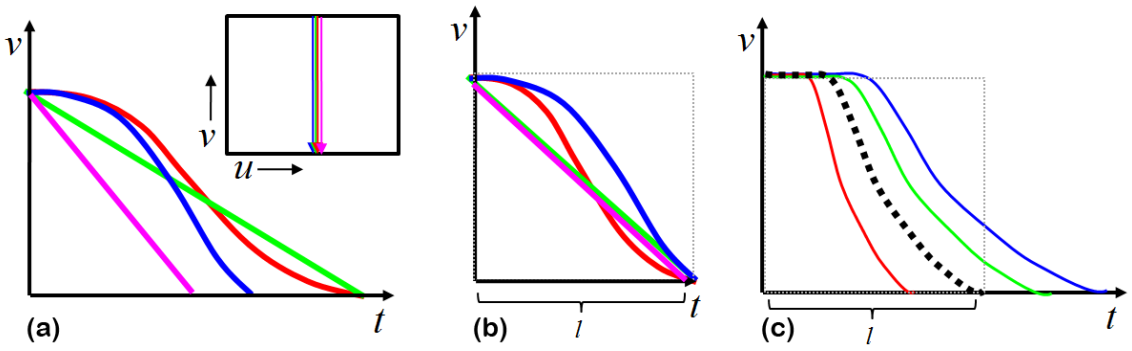


Figure 45: Normalizing frames: **(a)** Each red, blue, green and magenta line indicates the trajectories from tracks started from a same spatial position toward a same destination with different velocities, and accelerations (or in different frame rates). The image in the upper-right shows trajectories projected onto the spatial domain. **(b)** All of the trajectories are normalized in time (t) axis as l. **(c)** Another example showing that each colored line has the same trajectory with the same velocity and acceleration but diverges later. The dashed black line shows the normalized trajectory.

number of sample per trajectory.

Consequently, as shown in Fig. 45, all the GPRFs or trajectories are distributed in the same scale, and we can compare motion trajectories with same *re-sampled* scale regardless of their actual lengths and frame rates. We consider two tracks with different starting positions to be in different classes, even if they have the same set of velocities. However, one should choose an adequate number of samples from each normalized trajectory for constructing a GPRF for each class. In the next subsection, we describe some essential steps for the selection.

7.3.2 Number of Samples and Variances

The computed variances for each velocity component in Eq. 18 (i.e. $\mathrm{var}(y_u^*(x))$, $\mathrm{var}(y_v^*(x))$, $\mathrm{var}(y_t^*(x))$) depend on (1) the number of nearby samples and (2) the hyper-parameters for the selected covariance function used in each GP model. Unfortunately, the marginal log-likelihood objective is non-concave [109] and we can hope to obtain only locally optimal hyper-parameters unless good initial starting sets of

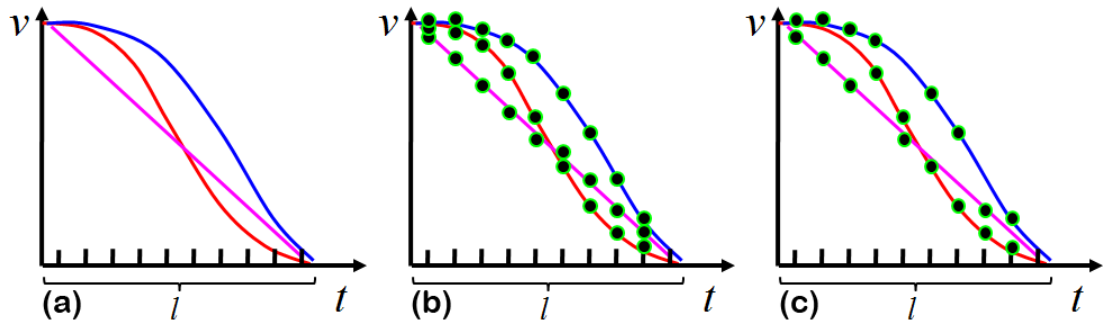


Figure 46: Sampling points for training: Suppose we are training a class using three trajectories. **(a)** All trajectories are normalized in l evenly distributed grid in time axis. **(b)** Adding all the samples per time grid for generating GPRF. In this case, training the same number of trajectories have to be guaranteed for all the other classes. **(c)** If we need only two samples per time grid, we randomly sample without replacement two points from three trajectories at each time slice.

hyper-parameters are provided. Therefore, balancing the adequate number of samples for each training set is a key element for stable GPRF learning. For Gaussian process regression (and GPRF consequently), the confidence band is narrower in regions with more samples. Therefore, comparing an input trajectory to each of the learned GPRFs, without the consideration of balancing, may be unfair, since it will favor the one with more samples. In addition, redundant observations and noise may affect the variances which would incorrectly quantify the level of *certainty*.

To tackle these problems, we first determine the number of samples per time step and allocate this same number of samples for each of the trained GPRFs. For each time step, we sample among the entire set of trajectories, which are chosen for a specific class, with random uniform probability and allocate samples among them. The random sampling method can alleviate both the unwanted reduction of variance due to the noise and the uneven distribution of variance levels in different classes. In our test, we use three samples per time step, as it gives us maximum confidence difference of 0.8 (worst case) in a resolution of our region of interest (the size of $u - v$ slice).

7.4 Similarity Measurement

We now have GPRFs computed for each class. Our next task is to evaluate the posterior probability density of a given online testing point x^o and its velocity vector y^o using Eq. 17 and 18 to measure similarity as a form of likelihood in the given training set (class). Let us first denote the online trajectory $\mathrm{T}_n = \{\mathbf{x}^o, \mathbf{y}^o\}$ as a trajectory with n observations, and the N–learned classes represented by GPRFs as $\mathbf{X}_k$ $(k = 1 : N)$. Assume that the time component of the online point is already normalized in unit lengths of l.

7.4.1 Measuring the Local Likelihood of Testing Data

As discussed earlier, we can evaluate a GPRF on any point regardless of the scale of the point. This evaluation is analogous to a prediction using the learned GPRF. In other words, given a GPRF class, the mean velocity vector in the location of testing point x^o is computed as: $\bar{y}^*_u(x^o)\mathbf{i} + \bar{y}^*_v(x^o)\mathbf{j} + \bar{y}^*_t(x^o)\mathbf{k}$ with the variances for each direction.

If the velocity vector of x^o becomes different, the posterior probability density will decrease upon the variance (confidence) of the point (See Fig. 47 (a), and (b)). If the position of x^o is further away from ALT, the variances for the velocity components become larger and result in a smaller posterior probability density (Fig. 47 (c)).

Suppose that we evaluate our online test point using the k–th GPRF, $\mathbf{X}_k$. Given an independent GP prior on each velocity component, we define the local likelihood of the testing point, χ on the training set $\mathbf{X}_k$, as:

$$\chi(x^o, y^o)|_{\mathbf{X}_k} = p(y^o_u|x^o, \mathbf{X}_k)p(y^o_v|x^o, \mathbf{X}_k)p(y^o_t|x^o, \mathbf{X}_k) \tag{19}$$

Note that the magnitude of y^o is normalized, and set to $|\bar{y}^*(x^o)|$ before the calculation. This likelihood is a main local similarity measurement in our approach. The details of how we use this likelihood in multiple classes of GPRF will be discussed in the following section.

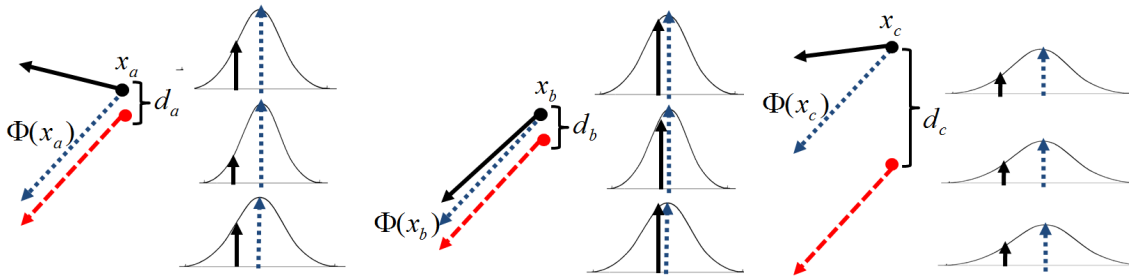


Figure 47: Measuring similarity with certainties: A red dashed arrow indicates the approximation of learned trajectory. A blue dotted arrow is a mean flow vector at each test point and a black arrow indicates its velocity vector. In the right side of each image, posterior densities for each u, v, t are shown. **(a)** The test point x_a is close to ALT, but the vectors are quite different. **(b)** Both the location of the input point x_b, and its velocity are close to mean flow and ALT respectively. **(c)** The example of the worst case: The test point x_c is further away from ALT (large variances of the posterior probability densities), and the velocity vectors are different (away from mean).

7.5 Classification and Prediction of Trajectories

In this section, we first describe a method to compute the global similarity between the trajectory and learned GPRFs. Using the local likelihood criterion (Eq. 19), we then describe methods for classifying patterns of trajectories, which can be either complete or incomplete. A *complete trajectory* consists of an entire set of segments collected from the moment a car enters to the moment it exits the observation region. Otherwise, a trajectory is *incomplete*. Incomplete trajectories are hard to classify because we do not know (1) when a trajectory ends (i.e. the car exits from the observation region) and (2) how the trajectory varies within the observation region.

7.5.1 Similarity for Complete Trajectories

Suppose that we have N GPRFs constructed from training datasets $\mathbf{X}_k (k \in [1, N])$, and a complete test trajectory $\mathrm{T}_n = \{x_i, y_i\}_{i=1}^{n}$. We define the global likelihood of an

input trajectory to be the *k-th* GPRF:

$$\mathcal{L}_k(\mathrm{T}_n) = \frac{1}{n}\sum_{i=1}^{n}\left[\chi(x_i, y_i)|_{\mathbf{X}_k}\right] \tag{20}$$

We also assign each part of the test trajectory $i = 1 : n$ to the class with the highest local likelihood:

$$l(i) = \arg\max_{k\in[1,N]}\left[\chi(x_i, y_i)|_{\mathbf{X}_k}\right] \tag{21}$$

We define c_k $(k = 1 : N)$ as the number of times $l(i)$ is equal to k for $i = 1 : n$. We can then compute the proportion of the input trajectory to be the *k-th* GPRF as $\rho_k(\mathrm{T}_n) = \frac{c_k}{n}$. The proportion is an effective parameter for the trajectory matching since it reflects how *dominant* the chosen class is. We refer to $\rho_k(\mathrm{T}_n)$ as a *proportion of dominance*. The final *global similarity* of the input trajectory to the specific GPRF k is then defined as:

$$\mathcal{S}_k(\mathrm{T}_n) = \rho_k(\mathrm{T}_n)\mathcal{L}_k(\mathrm{T}_n). \tag{22}$$

The index of the closest GPRF is $\arg\max_{k\in[1,N]}\mathcal{S}_k(\mathrm{T}_n)$.

While the classification procedure based on the proportion of dominance is similar to the weighted nearest neighbor approach, we demonstrate that knowing the distribution of ρ_k over all the classes gives us the notion of ambiguity. We will discuss this in Section 7.6. Fig. 51 shows a simple validation test of the above procedure.

7.5.2 Prediction from Incomplete Trajectory

We now assume that we are given an online trajectory of unknown length. Therefore, we have to predict the adequate scale for the incomplete track at time τ.

Let us first denote the function $\phi(x, \alpha)$, which scales only the third component (t axis) of the input point x by scalar α. We then evaluate Eq. 20 as:

$$\mathcal{L}^*_k(\mathrm{T}_\tau) = \frac{1}{\tau}\sum_{i=1}^{\tau}\left[\chi(\phi(x_i, \tfrac{l}{\tau+j}), \phi(y_i, \tfrac{l}{\tau+j}))\Big|_{\mathbf{X}_k}\right] \tag{23}$$

by varying k and j, where the l is a normalized length used for constructing GPRFs. If we take the j^* and k^*, which maximize the $\mathcal{L}_k^*$, the selected j^* gives us the right scale for the time axis, and $k*$ is an index of the chosen GPRF. Then, the percentage of local selection for $i = 1 : \tau$ can be calculated by counting c_k^* from the following function:

$$l^*(i) = \arg\max_{k\in[1,N]} \left[\chi(\phi(x_i, \tfrac{l}{\tau+j^*}), \phi(y_i, \tfrac{l}{\tau+j^*})) \Big|_{\mathbf{X}_k} \right] \tag{24}$$

From Eq. 22, we can then measure the global similarity as: $\mathcal{S}_k^*(\mathrm{T}_\tau) = \rho_k^*(\mathrm{T}_\tau)\mathcal{L}_k^*(\mathrm{T}_\tau)$. An incomplete trajectory often contains few vectors. Its similarity measure is thus incorrectly computed from a small number of earlier observations. To avoid this, we weight each term in $\mathcal{L}_k^*$ and each $l^*(i)$ in computing c_k^* to give larger weights to the testing vectors close to the current position at τ.

7.6 Anomaly Detection

Inspired by previous work on anomaly detection [24, 102, 54], we first define the properties of anomalous events in traffic motion trajectories and deal with the events from an unsupervised perspective while the normal patterns are labeled [54]. The properties of anomalous events we assume in this work (and our criteria to detect anomalies under the constraints) are the following:

Unlikelihood against normal patterns : Obviously, an anomalous trajectory is not likely to be learned from normal patterns (i.e. wrong direction, undefined pattern etc.). Our solution is to measure the global similarity of tracks to learned normal classes (Eq. 22), and see how much the similarity of the chosen class is far from the similarity from normal trajectories. Let us first denote the selected class as k, and its maximum global similarity as $\mathcal{S}_k(\mathrm{T}_n)$.

Ambiguity : Subsequences of a pattern sometimes fluctuate between normal patterns (i.e. zigzag patterns) or change from one class to another class. Even though we are not certain whether the subsequence goes into the category of anomaly or

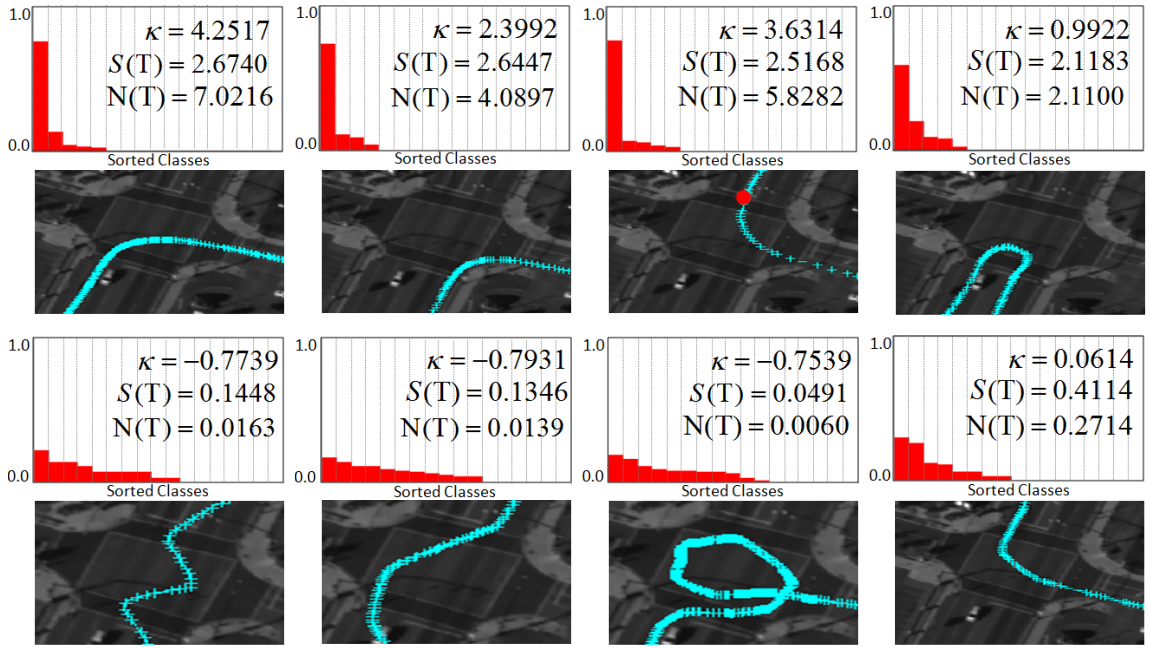


Figure 48: Kurtosis and Normality examples: Graphs in the first and third rows show the distribution of sorted $\rho_k(\mathrm{T}_n)$ of some examples. Images in the second and the fourth row are the corresponding trajectories. In each graph, y–axis denotes the proportion of dominance (0.0 to 1.0), and x–axis shows grids of 17 different classes (shown in Fig. 42). The leftmost bar in each graph shows a dominance level of the chosen class. Listed with each graph are the excess kurtosis κ, global similarity $\mathcal{S}_k(\mathrm{T}_n)$, and normality value $\mathrm{N}_k(\mathrm{T}_n)$. Our method classified the examples in the upper row as normal and lower ones as anomalous trajectories. Notice that the third example in the upper row models a car which *stopped* for a while at the red light.

not (since it was not labeled), the fluctuation and different selection of subsequences could increase the ambiguity for decision.

As discussed in section. 7.5.1, the proportion of dominance value $\rho_k(\mathrm{T}_n)$ effectively reflects this tendency. A distribution of $\rho_k(\mathrm{T}_n)$ with a broader spread represents a case of ambiguity in which there is no one dominant class. We interpret this as one of our criteria to detect anomalous events.

To verify this, we first sort the dominant proportion values $\rho_k(\mathrm{T}_n)$ $(k = 1 : N)$ in a non-increasing order. We then measure the *excess kurtosis* of this distribution. The excess kurtosis is calculated from $\kappa = \frac{\mu_4}{\sigma^4} - 3$, where μ_4 is the fourth moment of the mean from sorted ρ_k, and σ is the standard deviation. In any case, $-2 < \kappa < \infty$.

If $\kappa < 0$, the distribution of the sorted ρ becomes broader and the peak (dominance) becomes smaller. If $\kappa = 0$, the distribution is almost similar to normal distribution. We assume a trajectory with $\kappa \geq 0$ as more likely to be a normal.

Using the criteria above, we define the normality function N such that the online trajectory $\mathrm{T_n}$ to be a class k as:

$$\mathrm{N}_k(\mathrm{T_n}) = \tfrac{1}{2}\left[(\kappa + 2)\mathcal{S}_k(\mathrm{T}_n)\right] \tag{25}$$

Using the normality function, we first define m–trajectories, $\mathrm{T}^i_{\mathrm{n(i)}}$ $(i \in [1, m])$, used for training the class k, and the γ_k as the minimum normality value of evaluating all the m–trajectories to the class k. The decision criterion for the anomalous trajectory is then defined as $\mathrm{N}_k(\mathrm{T_n}) < 0.5\gamma_k$.

Fig. 48 shows examples of the normalcy test using some of our trajectory data using learned GPRF classes shown earlier in Fig. 42.

7.7 Results and Evaluation

We evaluate our method on four different video data sets taken from different locations (Fig. 49, and Table. 5). Since CLIF data does not have a sufficient number of trajectories, we added simulated trajectories generated from our traffic simulation software. Each data set has different frame rates and durations. Since our approach generating GPRF is not affine invariant, we rectified all the video sequences and used their trajectories in a canonical view (top-view). Each data set has more than 100 trajectories, which are labeled for testing purposes (Fig. 49, third column), and all have anomalous trajectories, except the UCF data. The UCF data set has a relatively shorter duration compared to the other data sets, and does not contain anomalous events. Each data set is divided into a training set, from which GPRFs are trained, and a test set for evaluation.

Similarity measurement of complete trajectory: We empirically evaluate the effectiveness of our method in trajectory classification by comparing to a well-known

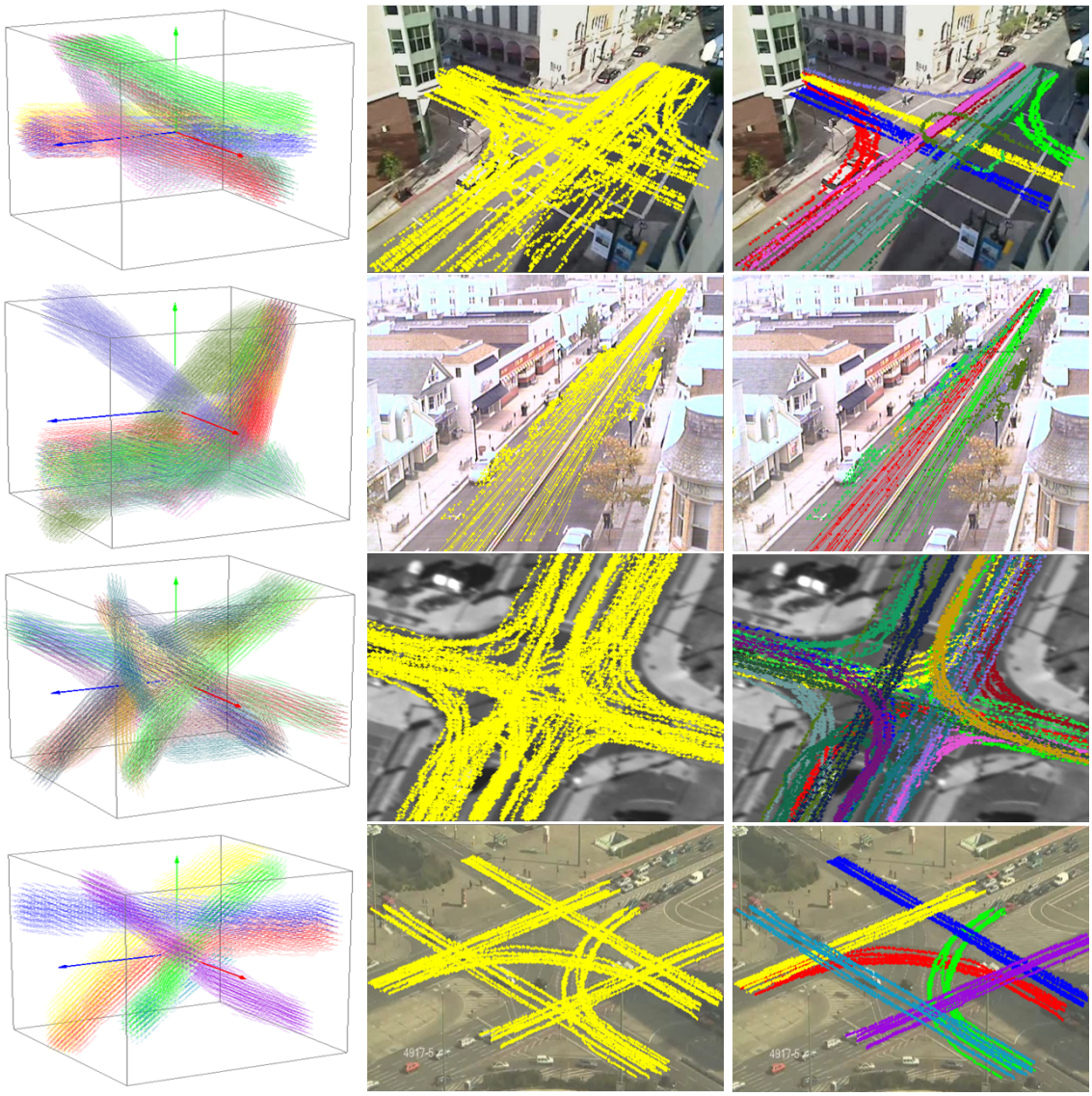


Figure 49: Datasets and each of their GPRFs: Example images from the videos used in this paper, and their learned GPRFs are shown. For each row, first column shows a normalized mean flows constructed from some of trajectories in each data set, and the second column shows testing trajectories, and the last column shows testing trajectories labeled in different colors for testing purpose.

trajectory matching algorithm. We consider the Derivative Direct Time Warping (DDTW) [61] method, because of its robustness in higher dimensional signals. For a fair comparison, we do not consider anomalous events. Table. 6 enumerates the average accuracy and precision of all classes using complete trajectories.

To evaluate the effectiveness in discriminating a specific pattern of trajectory from

Table 5: Dataset description

#GP refers to the number of trained classes (different types of straight, left-right turns, u-turns, stop-and-go, and parking with many of 2nd order movements) and ANO refers if there is anomalous sample exist in the data or not. For non-anomaly data set, we only evaluate the similarity.

Set	**Data type**	**#GP**	**frame rate**	**ANO**
Adam [145]	Intersection	8	10–30	Y
Ocean [96]	St.Road	11	5	Y
UCF [4]	Intersection	6	30	N
CLIF [133]	Intersection	17	varying	Y

Table 6: Trajectory recognition using complete trajectories

Each value denotes the average of each class. Our proposed approach works better in most of datasets except UCF data, which has small number of very distinctive motion patterns. See Fig. 49

	GPRF		DDTW	
Set	Accuracy	Precision	Accuracy	Precision
Adam	0.9955	0.9843	0.8928	0.7296
Ocean	0.966	0.8782	0.9488	0.7424
UCF	1.0000	1.0000	1.0000	1.0000
CLIF	0.9929	0.9488	0.9881	0.9274

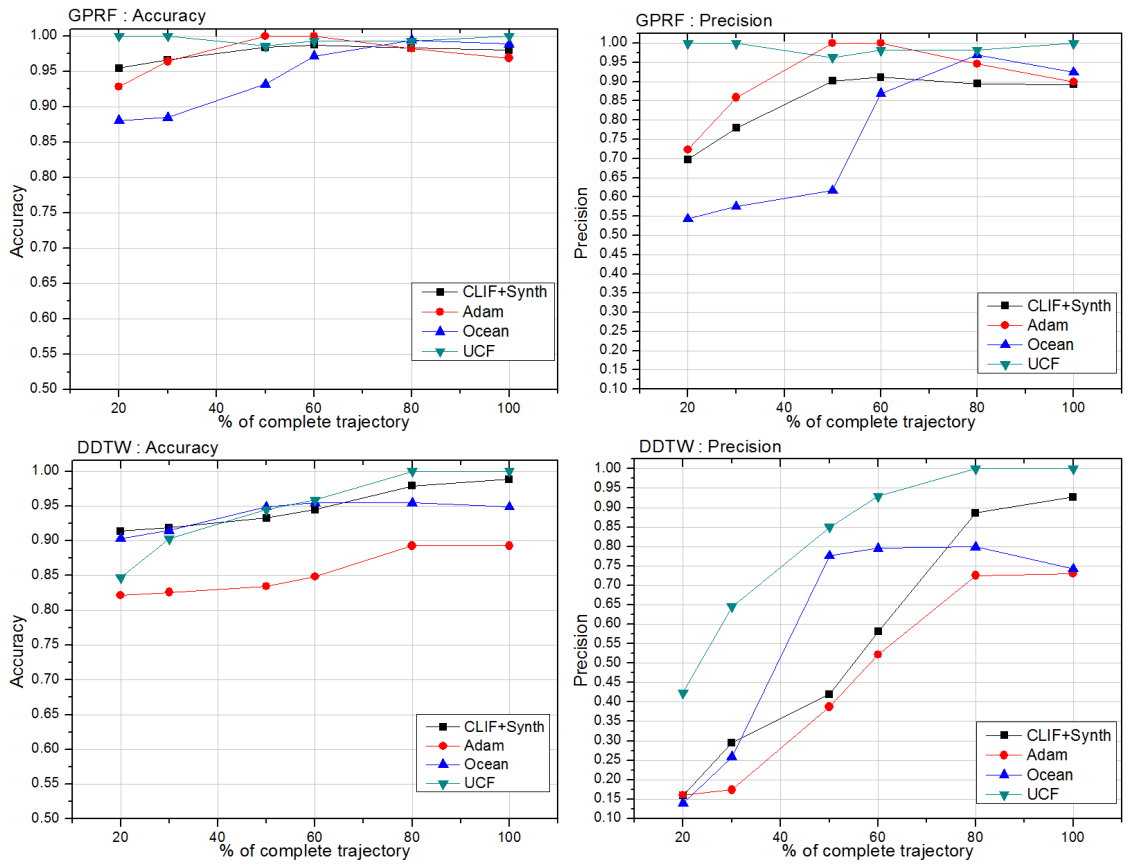


Figure 50: Trajectory recognition using incomplete trajectories: For each graph, x–axis indicates the percentage of sub-tracks used for evaluation (i.e. 50% means by the first 50% of total length of testing trajectory). (**Upper row**) Results from our approach, and (**Bottom row**) Results from DDTW

different patterns, we measure the global likelihood and the global similarity of randomly chosen trajectories from each trained class. Fig. 51 shows a comparison of the similarity measurements. Our global similarity measurement dramatically discriminates the input trajectories from ones belonging to incorrect classes and even from anomalous tracks.

Similarity measurement of incomplete trajectory : We evaluate our method on incomplete trajectories both quantitatively and qualitatively. The graph in Fig. 50 shows the quantitative result. Our approach works better in most of the data sets even with short duration of sequences from the input trajectory. Fig. 53 demonstrates how our approach works in incomplete trajectories from online video.

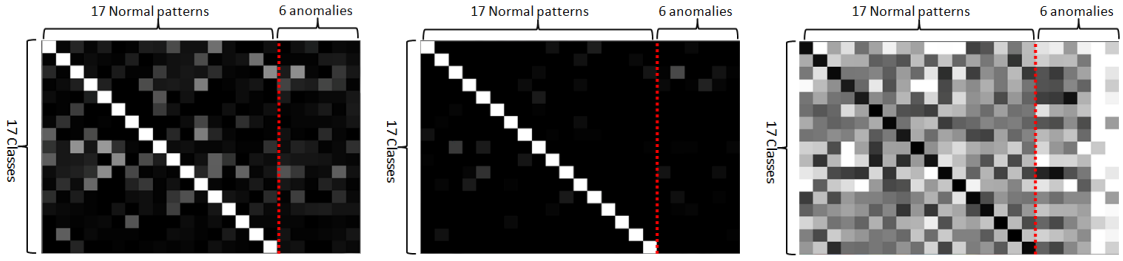


Figure 51: Similarity validation test: In each image, the row indicates learned 17 classes, and the first 17 columns are the trajectories which are randomly chosen from each of labeled sets (for the visualization purpose, we arranged each trajectory corresponding to the order of classes). The last 6 columns consist of randomly chosen anomalous trajectories from our anomalous sets. Note that all values are normalized from 0 to 255 by the maximum similarity of each classes. (Left) Using global likelihood (Eq. 20), (Middle) Using Global Similarity (Eq. 22). (Right) Minimum distances from DDTW : a darker color represents a better score.

Notice that the examples in Fig. 53 A and B describe how our approach can model and recognize complex motions such as deceleration and stopping situation, which is not possible in spatial proximity based approaches.

Evaluation of anomaly detection Fig. 52 shows ROC curves for each data set. Except the results from the Ocean data, other results show that TPR is more than 90% with less than 20% of FPR. As shown in Table 5, the Ocean data has very low frame rate, thus, its actual online trajectory is too sparse. Therefore, the kurtosis measurements were too sensitive, since the distribution were constructed from too few samples. However, it still gives us 80% of TPR with 30% FPR. Fig. 54 demonstrates that our method effectively detects anomalous events in incomplete trajectories.

7.8 Summary

In this chapter, we introduce a new framework for modeling motion trajectories in the spatio-temporal domain using flow fields generated from a GPRF. Our experiments demonstrate that our approach can recognize motion patterns from both complete and incomplete tracks even with GPRFs trained from a minimal number of labeled samples.

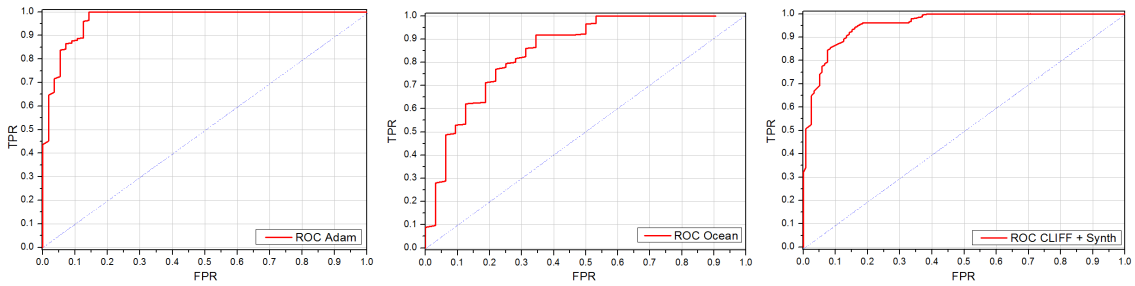


Figure 52: ROC curves for anomaly detection: x axis refers to False Positive Rate (FPR), and y axis for True Positive Rate (TPR), we evaluate each curve by varying a threshold on each normality function (Eq.25). **(Left) :** Adam, **(Middle) :** Ocean, **(Right) :** CLIF

There are some limitations and avenues of improvements. First, our approach assumes that the types of normal patterns are defined *a priori*. Secondly, if the trajectory is too sparse, the kurtosis measurements in earlier stages of the track can be unstable due to the sensitivity of kurtosis in a sparse distribution. Finally, our approach does not recognize traffic jams and such patterns would be detected as an anomaly. A traffic jam, however, can be represented as a *step* shape function in the normalized (u, v, t) coordinates, and we plan to use this method to model traffic jams.

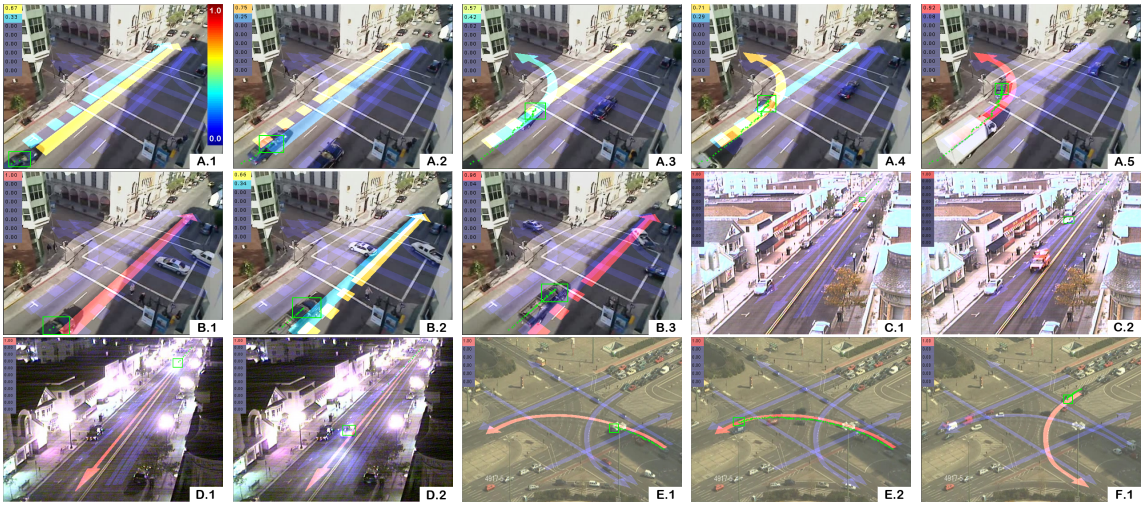


Figure 53: Qualitative evaluation of trajectory prediction: Each arrow indicates a possible pattern of motion on the road. The color of each arrow indicates the probability level to be classified as the given pattern (See the color table in A.1). Dotted lines indicate the "stop (includes deceleration) and move" **A.1**. A car moves straight ahead with a probability of 67% for continuing to move straight. **A.2** As the car decelerates, a pattern for stopping situation becomes dominant. **A.3** The car slightly changes its direction to left. Because it needs to decelerate before a left turn, the probability of taking the left turn increases. **A.4** The left turn becomes dominant. **A.5** Finally, even before the car enters the left road, the probability for a left turn increases to 92%. **B.1** A car originally moves ahead in the second lane with the probability of 100% for continuing that motion pattern. **B.2** The car is slowing down. **B.3** Finally the car stopped for a traffic signal. **C.1** A car moves ahead to bottom left, but decreases its speed. Therefore even the spatial pattern is similar to straight, the patterns for parking becomes dominant **C.2** Finally the car parks. **D.1–2** Though spatial movement is similar to the examples shown in C, due to its velocity and acceleration, it was proven to be straight pattern. **E.1–2** and **F.1** Examples of prediction in UCF data

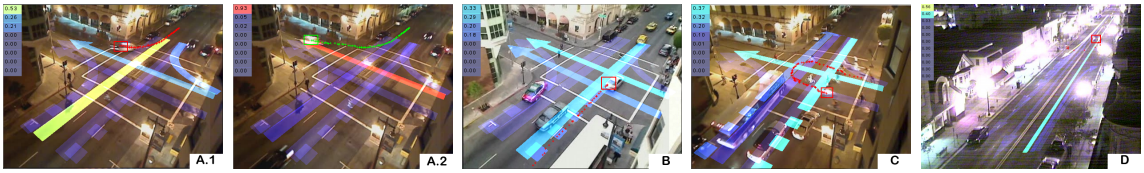


Figure 54: Qualitative evaluation of incremental anomaly detection: The color of the trajectory indicates the normalcy (green), and anomaly (red) **A.1** A police car goes in the wrong direction on a one-way road, and is classified as an anomalous pattern. **A.2** As the car merges to left direction, the trajectory becomes normal. **B** A car was suddenly stopped inside the intersection due to a pedestrian, and we classify it as an anomaly. **C** Bikes are moving inside the intersection. **D** A car takes a U-turn which is not allowed in the street.

CHAPTER VIII

STOCHASTIC APPROACH FOR GLOBAL MOTION ANALYSIS IN DYNAMIC SCENES WITH CAMERA MOTION

In Chapter 6, we introduced an approach to measure the global tendencies from sparse set of motion with radial basis function interpolation, and detect the location in which the global motion merges. We interpreted the location (point of convergence; POC) as a location where the observing camera has to move. However, the approach depends on well configured static-multi-view videos for the precise motion samples on the ground, and may not be an adequate solution for a practical scenario where a PTZ camera is able to both analyze and move itself.

In order to model such configurations, in this chapter, we propose a method to predict the point of convergence by constructing a stochastic vector field with Gaussian process regression. The vector field is built on the regression model and covariance function that are modeled with residual terms, and all the motion vectors at any location can be represented by a set of means and variances, as we already discussed in Section 4.2. The means collectively reflect the motion tendency, whereas the variances quantify the level of confidence of the motion tendency as the motions may be sparse and noisy. From the generated stochastic vector field, we predict the location where the camera moves even from the videos with camera motions. Fig. 1 shows an example of the change of view over time as the camera operator pans and widens the camera view during a football play. The top figures show the two frames with a overlapping region from left to right. The bottom figure lays out the image on the football field and shows a stochastic motion field on the ground generated using

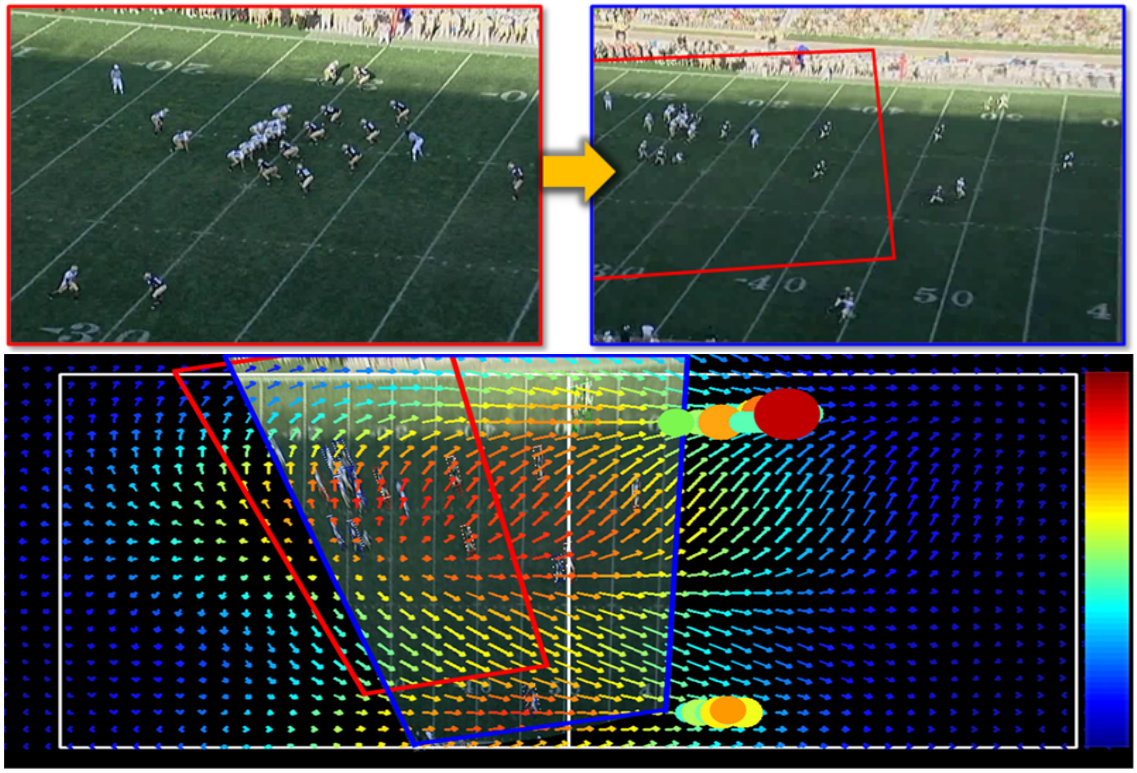


Figure 55: Overview: **Top:** An example of the pan-tilt-zoom performed by a camera operator. The field of view changes from the region bounded by red lines to another bounded by blue lines, **Bottom (overhead view):** Arrows indicates a motion field generated only from the ground-motion of players with Gaussian process regression. The certainty level of the velocity vector at each location is represented by different colors. Circles indicate the predicted future locations of interest which can be interpreted as the location where the field of view of the camera will move. Color bar represents the level of values (Red denotes higher values).

Gaussian process regression, which allows for computing of how the play is unfolding, as the players are tracked and followed over the field to determine regions of interest (POC).

Our contributions in this Chapter are: (1) A method to generate a stochastic motion field that represents a global motion tendency using Gaussian process regression (GPR). (2) Techniques for predicting important future locations from mean and variance fields computed from the stochastic vector field. (3) An evaluation method for measuring the "goodness" of predicted important regions. We utilize the Jaccard coefficient [127] computed from the fields of view covered by our approach and *actual*

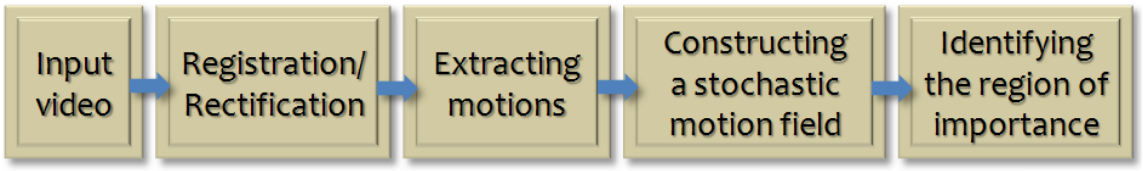


Figure 56: From input video to the detection of future important locations

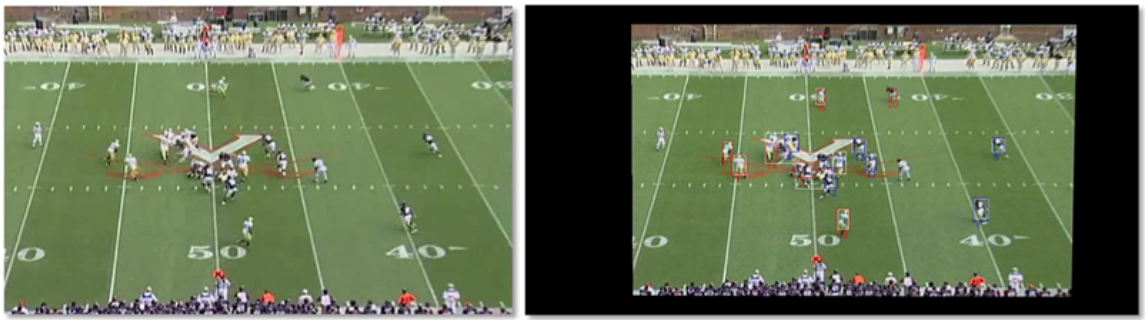


Figure 57: Registration of each frame onto the reference view for a player tracking: **Left:** Original view, **Right:** A rectified view used for tracking players

camera operators (the base-line comparison). The Jaccard coefficient is an intuitive similarity measure; the size of the intersection divided by the size of the union of two sets. Based on our criterion, we demonstrate that our approach can predict regions of importance quite accurately. We demonstrate the validity of our approach over a very complex data set, which will be made available with the paper.

8.1 Detecting Regions of Interest

To automatically capture the dynamic sports scenes by identifying where the global motion tendency moves, we first extract motions on the ground plane in the scene. We then generate a stochastic motion field for representing the global motion tendency. Finally, we detect the locations where the motion field converges. Figure 56 shows the overall framework.

8.1.1 Computing Motion on the Ground

We first register video frames into the known field coordinates using successive local features appearing in each video frame [51]. Then we rectify video frames into a

reference frame with successive homographies extracted from the registration. We chose the first frame of each video clip from the data sets as a reference for the registration (Figure 57 (Right)). This rectified video frame is used for extracting the ground motion of each player by applying particle-filter based tracking [72]. We then approximate the motion vector on the ground as the vector between the center of bottom edge of tracked blobs in each consecutive frame. In order to construct the motion field from the extracted motions on the ground, we project all the motions into the overhead-view of the ground field as shown earlier in Figure 55 (Bottom).

8.1.2 Stochastic Motion Field

Our task is to generate a dense motion field representing global tendency with sparse motion extracted from the scene. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Re^d$ be a set of locations of extracted motion, in which d is the dimension of the input motion that we want to model. Each location $\mathbf{x}$ has a set of noisy observed velocity vector components: y_u (the velocity component in the u-axis), y_v (the velocity component in the v-axis), (and optionally y_t for modeling the component in the time-axis). We assume that each velocity component at the location $x \in \Re^d$ follows the regression model $\hat{y} = f(\mathbf{x}) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, *i*.e., Normal distribution.

Gaussian Process Regression. We propose using the Gaussian process regression model, where $f(x)$ is a zero-mean Gaussian process with covariance function $K(x, x'') : \Re^d \times \Re^d \rightarrow \Re$. A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [108]. It is completely specified by a mean function $m(x) = \mathbb{E}[f(x)]$ (typically assumed to be $\mathbf{0}$) and a covariance function $K(x, x'') = \mathbb{E}[(f(x) - m(x))(f(x'') - m(x''))] = \mathbb{E}[f(x)f(x'')]$.

If we have training data $D = \{x_i, y_i\}_{i=1}^N$, the $N \times N$ covariance matrix $\mathbf{K}$ is now defined as $[\mathbf{K}]_{jk} = K(x_j, x_k)$. We then define the observation vector $\mathbf{y} = [y_1, \ldots, y_N]^T$; $\mathbf{y}$ can be shown as a zero mean multivariate Gaussian process with a covariance matrix

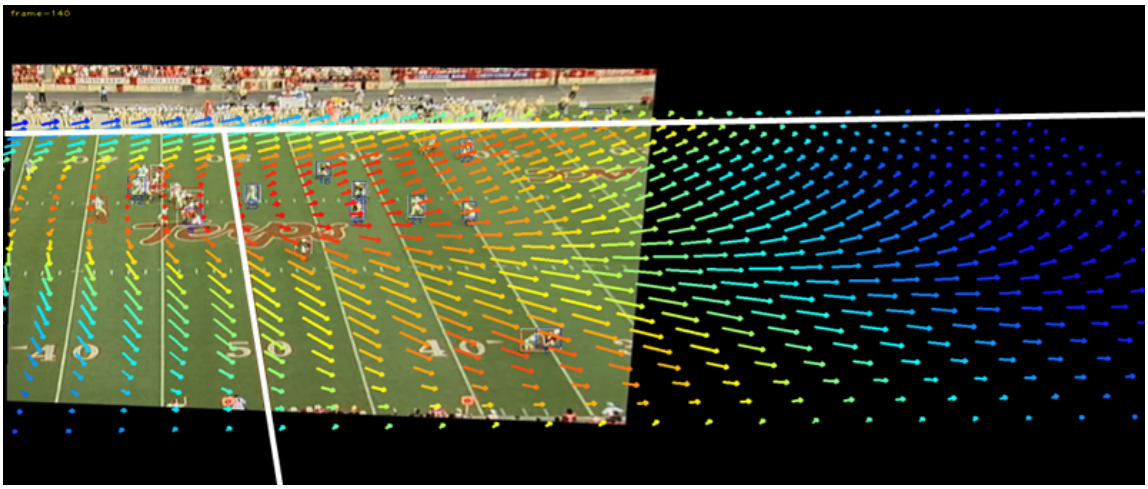


Figure 58: Stochastic motion field and its certainty field generated from GPR: The arrows indicate the vectors in the field generated from the motions of players. The colors of the arrows represent the level of certainty. Red arrows have larger certainty level (narrower confidence band) and blue ones have lower certainty level. Therefore extrapolated vectors are more likely to be blue. Bold white lines indicate the references for the ground plane. Note that all calculations were done in the overhead projection.

$\mathbf{K}^* = \mathbf{K} + \sigma^2\mathbf{I}$. The posterior density for a test point x^*, $p(y^*|x^*, D)$ is a univariate normal distribution with the mean $\bar{y}^*$ and the variance $\mathrm{var}(y^*)$:

$$\bar{y}^* = \mathbf{k}(x^*)^T(\mathbf{K}^*)^{-1}y$$

$$\mathrm{var}(y^*) = K(x^*, x^*) - \mathbf{k}(x^*)^T(\mathbf{K}^*)^{-1}\mathbf{k}(x^*)$$

where $\mathbf{k}(x^*) = [K(x^*, x_1), \ldots, K(x^*, x_n)]^T$. We can then express the mean flow as a vector field for two dimensional motions, $\Phi(x) = \bar{y}^*_u(x)\mathbf{i} + \bar{y}^*_v(x)\mathbf{j} \in \Re^2$ (u, v represent a spatial domain), and for three dimensional motions as $\Phi(x) = \bar{y}^*_u(x)\mathbf{i} + \bar{y}^*_v(x)\mathbf{j} + \bar{y}^*_t(x)\mathbf{k} \in \Re^3$ (t indicates a temporal domain) with a variance for each velocity component $\mathrm{var}(y^*_u(x)), \mathrm{var}(y^*_v(x)), \mathrm{var}(y^*_t(x))$ respectively. Figure **??** shows the generated stochastic motion field using mean field $\Phi(x)$. In the figure, the certainty level is shown as color values indicating a 95% of distribution (confidence band) with the mean and variances in each velocity of the motion.

Comparison with the Deterministic Motion Field. We note that the mean

of the posterior density $p(y^*|x^*, D)$ can be interpreted as a linear combination of observations $\mathbf{y}$, *i*.e., a *linear predictor*: $\bar{y}^* = \sum_{y_i} \alpha_i y_i$ where $\alpha = (\mathbf{K}^*)^{-1}\mathbf{k}(x^*)$. At the same time, it can also be written as: $\bar{y}^* = \lambda^T \mathbf{k}(x^*) = \sum_{x_i} \lambda_i K(x^*, x_i)$ where $\lambda = (\mathbf{K}^*)^{-1} y$, *i*.e., a weighted kernel sum that appears in the RBF approach shown in [67]. The GPR approach is similar to the RBF, except for one key difference. RBF is used primarily for interpolation of known observation values, whereas GPR can operate on noisy observation values. By specifying the mean function $m(x)$ and the covariance function $\mathbf{K}(x, x'')$, we can construct confidence bands around each predicted value to quantify the certainty level. Readers can refer to [8] for more details on some key similarities and differences between the RBF and GPR techniques. In our experiments, we validate the effectiveness of confidence bands for identifying convergence locations of motion trends.

8.1.3 Detecting Locations of Convergence

Detecting convergence locations of the motion field can guide the movement of the PTZ camera as shown in [67]. The approach propagates and updates a magnitude of a velocity along a motion field. However, this approach often suffers from two problems. First, propagating every vector (regardless of its similarity to actual global motion tendencies) in the field has been shown to be computationally intensive. Secondly, extrapolated velocity vectors with large magnitudes can seriously bias accumulation and yield an unstable localization of converging points.

We make several modifications to address the issues listed above. First, instead of propagating a magnitude of velocity, we *transport* certainty levels computed from GPR. Second, we transport the certainties only for the locations with high certainty levels. We note that confidence bands predicted from GPR are wide for locations that are far away from actual input motion vectors; these locations with extrapolated velocity vectors are otherwise unnoticed under the RBF computation [67], but can

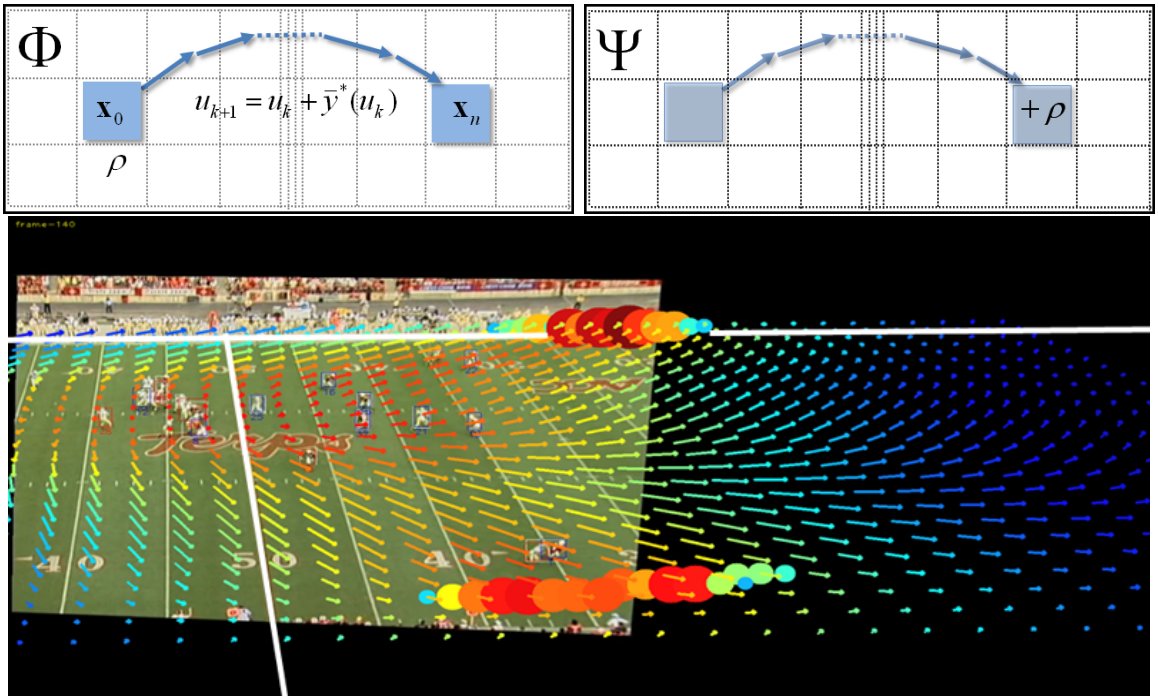


Figure 59: Certainty transfer through stochastic motion field and merging points: **Top-left:** The certainty level ρ at a location $\mathbf{x}_0$ is transferred to the location $\mathbf{x}_n$ through the stochastic motion field Φ. **Top-right:** In a separate grid Ψ, the value ρ is accumulated at the location of $\mathbf{x}_n$. Accumulated certainties in Ψ will be used to predict locations of future importance. **Bottom:** Colored circles indicate accumulated certainties from the motion field shown in Figure **??** (red circles with larger accumulations and blue ones with smaller accumulations). Note that we visualized the only locations that have more than 80% of maximum accumulation.

be accounted by our new approach. Third, transporting the certainty level requires updating only the last destination point and is computationally more efficient.

We first define an evaluating function $\mathcal{E}(n, \mathbf{x}, \Phi)$, where Φ is a motion field (mean field), $\mathbf{x}$ is a starting location, and n is a number of the iteration. Starting from $\mathbf{x}$, $\mathcal{E}(n, \mathbf{x}, \Phi)$ follows the flow of Φ by integrating predicted velocity vectors at each iteration. For example, if we denote the location of a motion as $\mathbf{x}_i = [u_i\ v_i] \in \Re^2$ $(0 \leq i \leq n)$, the evaluating function $\mathcal{E}(n, \mathbf{x}, \Phi)$ iterates the locations by $u_{k+1} = u_k + \bar{y}_u^*(u_k)$, where $y_u^*(u_k)$ is computed from $\Phi(\mathbf{x}_k)$. We denote the final location returned by the function after n iterations as $\mathbf{x}_n$ (See Figure 59). Through this function, we transport the certainty value ρ (95% of confidence, $1.96\sigma^2$) from $\mathbf{x}$ to $\mathbf{x}_n$ along the field Φ.

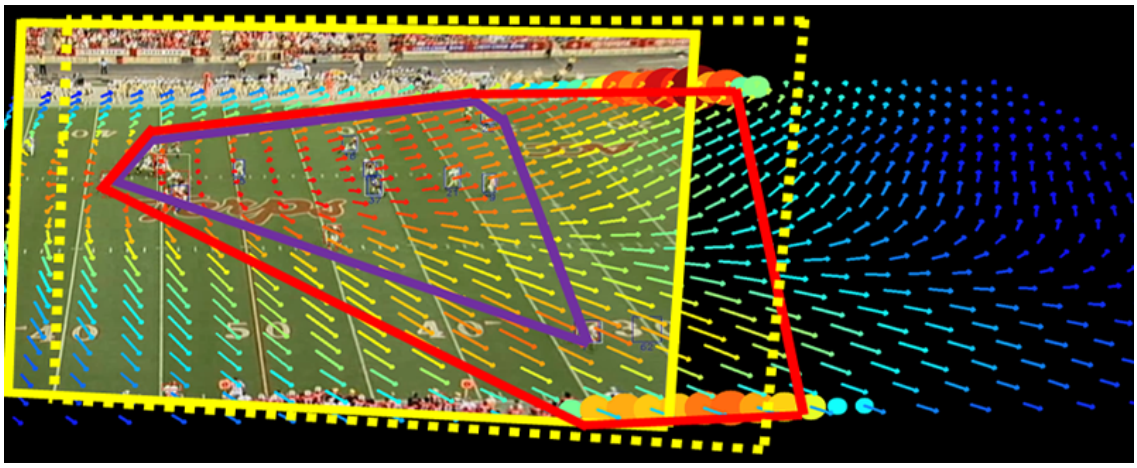


Figure 60: Evaluation for the comparison of actual camera operator's field of view: The region with solid yellow lines denotes the camera operator's field of view, whereas the region with dotted yellow lines represent the field view 10 frames later. The convex hull of only the locations of players is shown with purple lines. The region with red lines is decided by the locations of players and the merging points computed by GPR.

Therefore, we only add the certainty values at the destination. We evaluate $\mathcal{E}$ at each position in the field with velocity vectors with sufficiently low variance. Figure 59 shows the resulting accumulated distribution of certainties Ψ from the original field Φ. In the following sections, we denote the locations with the values accumulated more than 50% of maximum accumulated values in the field as merging points.

8.1.4 Measuring the Similarity with Field of View

We use the region of the field of view controlled by real camera operators as the baseline comparison (see Figure 60). For each approach to be compared against the baseline, we first construct a convex hull formed by the locations of players and convergence locations detected by the approach. The similarity metric between the constructed convex hull region of an approach and the baseline field of view is chosen to be Jaccard coefficient. We measure the similarity between the field of view decided by camera operators and the region of the convex hulls in the evaluating frame. We repeat this evaluation for each successive pair of the baseline field of view and the

convex hull region for an approach as shown in Figure 61 (Top).

There are several reasons for this evaluation; first, we want to validate our hypothesis that the predicted global motion tendency and its merging points (in addition to the locations of moving objects) can be used to adjust the field of view. Second, we want to verify whether the prediction based on motion tendency is similar to the field of view adjusted by actual camera operators. We note that the evaluation of the second criterion is not possible under multiple static-view camera approaches. Finally, we want to verify whether the predicted regions of importance (represented as player locations or merging points) computed from each method can be readily deployed *without* additional post-processing methods. The examples of such post-processing methods include a bounding rectangle with margins [10] and linear camera motions [67], which are not suitable for automated *live* application.

8.2 Evaluation and Results

For our experimentation, we have worked with two different data sets. First, we use American football video data taken by real camera operators; the videos are taken from moving cameras and have various PTZ changes. The data set consists of 8 different video clips (cv1 - cv8) from US College football games. This data set is used for the comparison of existing algorithms to actual camera operators' view-adjustment. In addition, this evaluation will measure the similarity between the predicted camera movement by each approach and the actual camera operator's decisions. Secondly, we also use video data sets from static-multi-view videos to compare our approach with existing methods used for static videos. The data sets consists of several videos from a soccer game used in Chapter 6 ([67]).

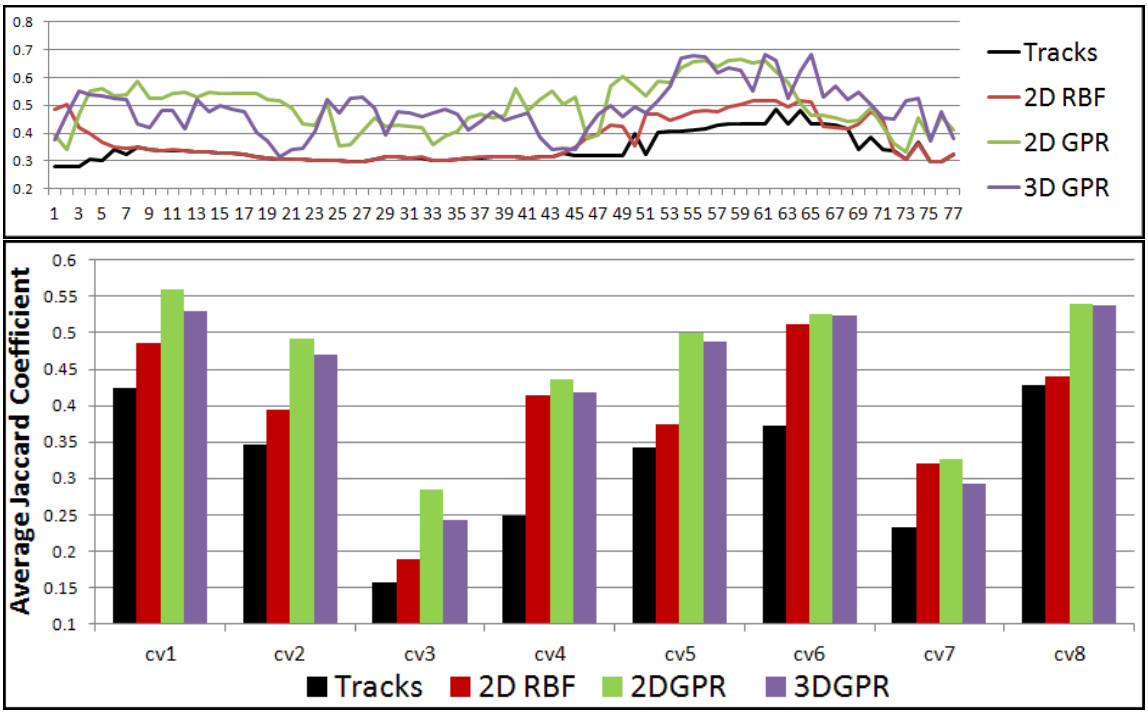


Figure 61: Quantitative evaluation for the comparison of actual camera operator's field of view: The values in vertical (y)- axis in both graphs indicates a Jaccard coefficient between a camera operator's region and each computed region, which uses the location of players (black), the 2D RBF (red), the 2D GPR (green), and the 3D GPR (violet) respectively. A graph in the top shows the evaluation over all frames from one sample from our data set (cv5). The bottom graph shows the average of Jaccard coefficient for all the football data sets (cv1 to cv8).

8.2.1 Similarity with actual camera operator's view

Graphs in Figure 61 show the average similarity using Jaccard metric between camera operator's field of view and the region decided by each method (1.0 is ideal). As expected, methods using the prediction based on motion field outperformed the method using only tracked results on every data set. Among the methods, 2D GPR-based prediction was better than RBF-based approach, and was even slightly better than 3D GPR-based approach in most of the cases. While the 3D GPR approach was shown to be an effective way to represent 2nd order movement of motion, and to be useful for motion recognition [68], the temporal derivatives do not seem to play an important role for discovering global motion tendency to identify the *location* of

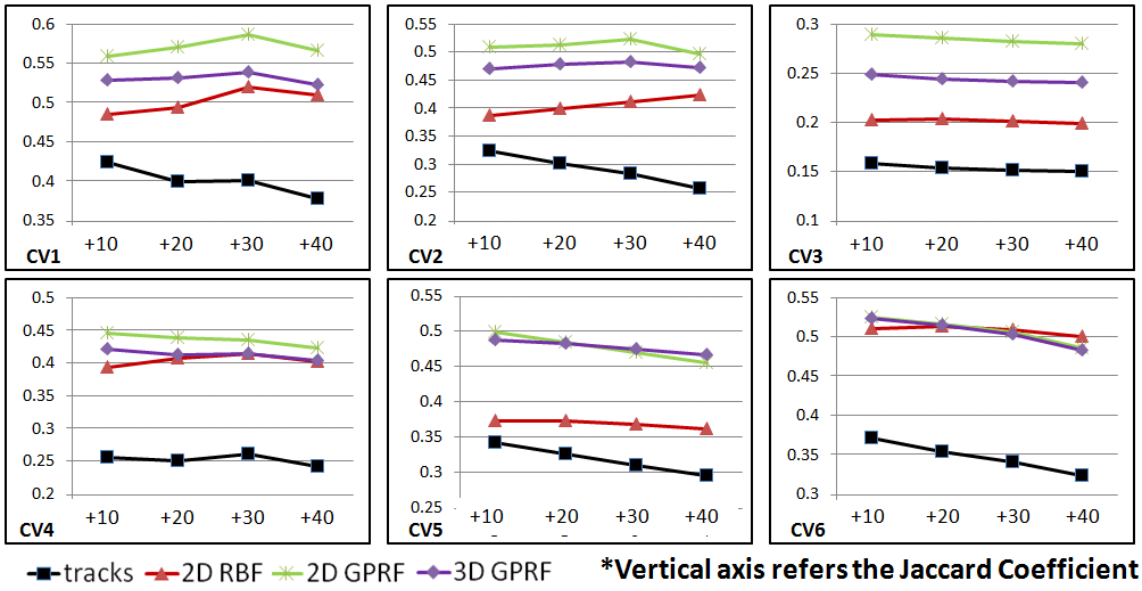


Figure 62: Evaluation of the future region of camera operator by differing frames: We evaluated each data set (cv1 to cv6 are shown here) by comparing the field of view of camera operator in future frames with the predicted merging regions at current frame by differing the frame difference from +10 to +40 frames. As shown in each figure, the method using only tracked player locations has lower similarity, and is generally decreasing as it always stay at the current location of players while methods using motion field have larger similarity over different frame offsets. In the results from cv4 and cv6, both GPR and RBF methods give similar results, while the other data sets show that GPR-based approaches work better. In the two data sets (cv4 and cv6), because the actual region of interests are close to boundary and fewer extrapolated vectors are involved in the prediction, both GPR and RBF methods give similar results.

importance, as projected 2D tendency has sufficient representation for the task.

The GPR-based approach generally outperformed the RBF method. Unlike the RBF interpolation-based approach, the GPR-based approach provides confidence bands at each velocity vector. Using these confidence bands, we can selectively propagate certainty values, whereas RBF interpolation requires iterating over every velocity vector. As a result, the GPR-based approach is less affected by (1) extrapolated velocity vectors than the RBF-based approach, and (2) errors present from registration and tracking of the original video data.

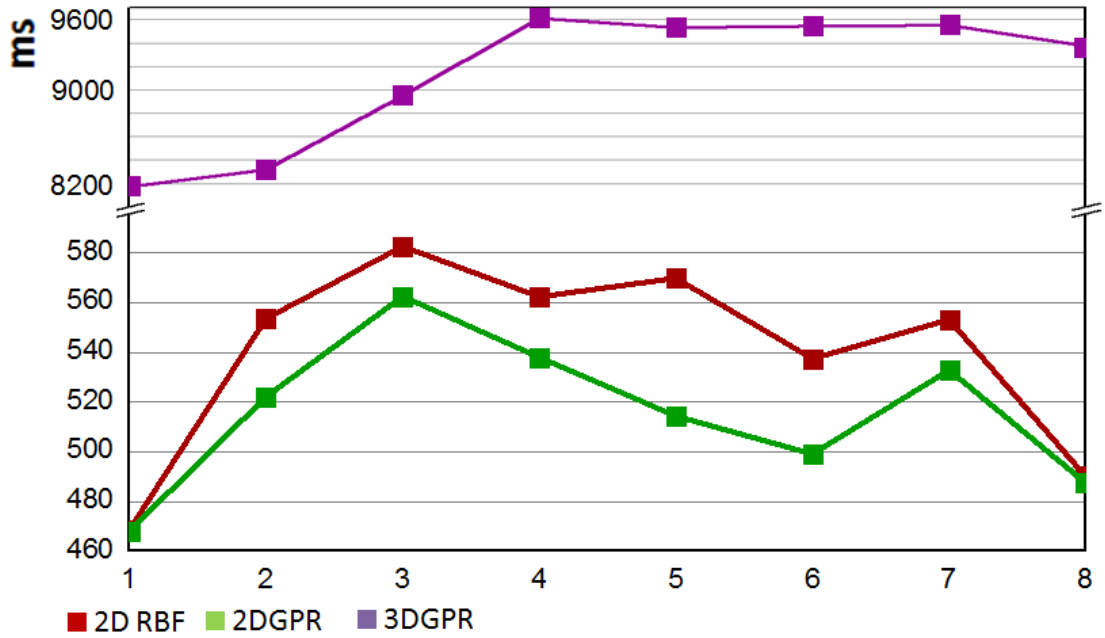


Figure 63: Computational expense: Red line indicates the computational expense of 2D RBF-based approach. Green indicates those of 2D GPR-based approach, and violet line indicates those from 3D GPR-based approach. x axis refers the each data set, and y axis refers the millisecond.

Figure 62 describes a more reasonable comparison in which we compare the region computed from each method with future regions adjusted by camera operators by differing frame offsets from 10 to 40. This evaluation provides a notion of how each method *foresees* the important regions in the scene. 2-D GPR-based approach outperforms the other approaches including the method using only tracking information. In addition, because the tracking-based approach uses only the current locations of moving objects, its effectiveness in directing the camera movement (shown via the Jaccard coefficient) drops markedly as the frame offset is increased from 10 to 40.

8.2.2 Computational Expense

Figure 63 shows the computational expense to perform the RBF-based and GPR-based methods. Basically, the main bottleneck for both methods are the summation of weighted values in the n by n kernel, which also requires matrix inversion. Even

though the GPR-based method has an additional computation for evaluating certainty levels, this evaluation is trivial when we consider a step for detecting the region where global tendency merges. This is because while the RBF-based method propagates (and updates) all the vectors followed by the motion field, GPR method transfers and updates only the final destination by excluding the extrapolated vectors having low certainty level. Therefore, the overall computation needed for the entire framework for the GPR method is faster. The 3D GPR-based method requires the formation of kernel matrices of size n^2 by n^2 (whose number of elements are n^2 times of kernel matrices formed for the 2D GPR-based approach). The 2D GPR-based approach is not only more computationally efficient but also can generate qualitative and quantitative results similar to those generated by the 3D GPR-based approach.

8.2.3 Qualitative evaluations for both moving and static cameras

Figure 64 demonstrates qualitative results from both RBF-based and GPR-based approaches in the video from moving cameras. As shown in the figure, our 2D GPR-based approach can predict regions of interest similar to ones implied by the camera movement. On the other hand, the merging points computed from RBF are commonly located near boundary regions because of the portion of extrapolated motions involved in the detection. Figure 65 showcases two resulting examples from our data sets (using 2D GPR). The detected merging points from both examples reasonably describe the motion of the actual camera.

To give additional comparisons with existing methods in static-view, we also applied our approach to the data sets captured from multiple-static cameras, which were used in Chapter 6 ([67]). Figure 66 shows some qualitative results showing the comparison between our approach and the RBF-based method. As shown in each sequence in the figure, the results from both approaches (RBF and GPR) do not look too different unlike the test using moving camera. First, because the motions on the

ground are relatively stable (as they are extracted from static multi-view), the error handling in GPR does not play an important role. Second, as the region covered by multi-view is smaller (just half the field with specific boundary conditions) than the data sets from moving cameras, there are fewer extrapolated vectors in the scene. Therefore, a velocity propagation without filtering the extrapolated velocity may be enough for identifying the merging locations.

8.3 Conclusion

We have shown that the prediction of the region of interests from stochastic field using Gaussian Process Regression provides robust results even with noisy motions from moving cameras. We demonstrate that the GPR-based approach can model the camera motion performed by actual camera operators more closely. In our future work, we will work on (1) improving the scalability of the code-base by utilizing GPGPU-based acceleration since most computations consist of matrix-matrix products (an embarrassingly parallelizable primitive); (2) applying our approach for controlling actual robotic cameras in real-time.

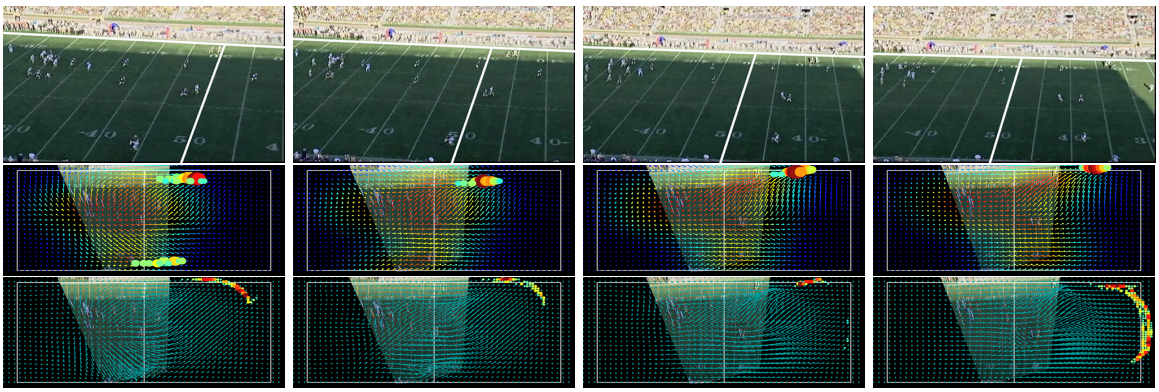


Figure 64: Qualitative evaluation between RBF method and GPR method: Top row shows the transition (PTZ) of the original views adjusted by the camera operator. To give a better understanding of how the original view moves, we added white lines to represent the 50 yard line and the upper boundary of the ground field. The view is being panned to the right direction, and zoomed out. **Middle row** shows the registered over-head projection of the stochastic motion field, and merging points computed from the 2D GPR method. **Bottom row** represents the result from the RBF based approach. Note that the merging points in RBF method are often concentrated near the boundary of the field because the computation of the merging points are highly affected by the extrapolated vectors in RBF (see the last example of the third row).

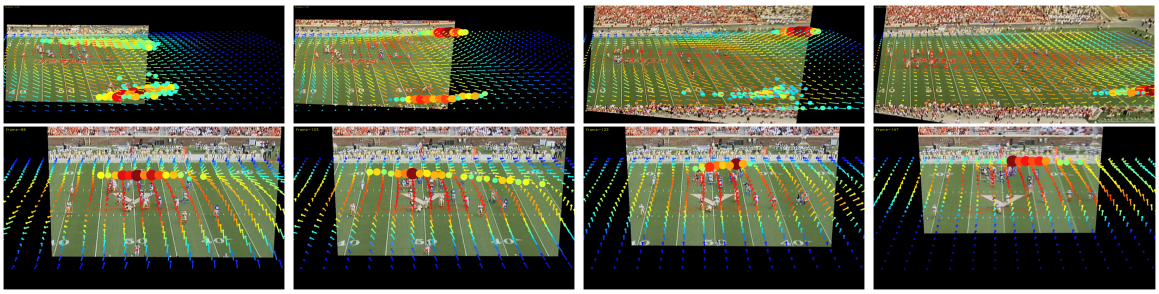


Figure 65: Additional comparison of our results (with GPR method) and actual camera motion: Sequences in **Top row** demonstrate how our approach mimics pan and zoom-out. For the sequence at **Bottom row**, as merging points staying at the center of field of view, then move away from the camera operator, the field of view changes from panning to zoom-in.

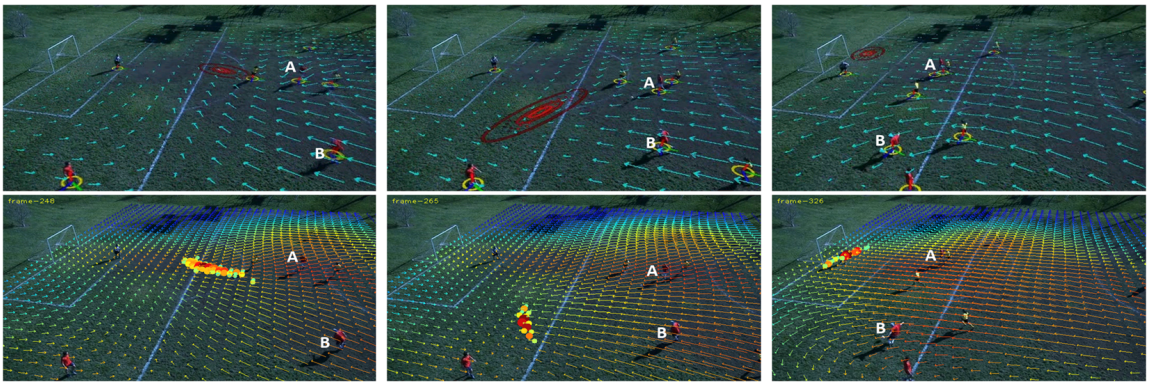


Figure 66: Qualitative comparison between RBF method and 2D GPR method: The sequence of scenes in **top row** show the result from RBF method (Chapter 6); the red contour indicates the location where the motion field merges. The scenes in **bottom row** show the result of our approach using GPR. The location of where the motion field merges is shown with circles in which the colors represents the amount of accumulated transferred certainties. For each row, first scene describes the merging location lies in front of player **A** who dribbles the ball. In the second scene, merging location describes the location where the other offender **B** will receive the ball. In the last scene, results shows the location for the other pass.

CHAPTER IX

SUMMARY AND CONCLUSION

In this thesis, we address the incomplete data problems caused by the availability of sparse spatio-temporal data in dynamic scene analysis and visualization. We hypothesize that the prediction of motions from scattered data interpolation and approximation (SDI/SDA) would be an effective solution. Then we show that applying SDI/SDA in a spatio- temporal domain can solve various types of sparse data problems in several applications with video analysis. In Chapter 4, we also introduced several well-known scattered data interpolation/approximation methods and described how we could adopt these methods into the spatio-temporal domain in dynamic scenes.

To visualize the dynamic scenes of wide areas (e.g., city visualization) with a sparse number of videos taken from cameras in different locations, we showed in Chapter 5 that a variety of interpolation techniques could be used in various camera conditions. Among the conditions, we showed that radial basis function (RBF) interpolation played an important role in propagating motion information extracted from observations of locations close to the unobserved regions. We also showed that data interpolation performed with prior knowledge of a given scene and an assumption of a behavioral model of moving objects could also be another solution to visualize wide dynamic scenes even with cameras with a limited field of view.

To effectively adjust the field of view of a pan-tilt-zoom (PTZ) camera used in surveillance and broadcasting applications, we proposed a method of calculating the global motion trend from sparse motions using RBF interpolation in Chapter 6. From the motion trend, we were able to calculate the region of importance using forward

velocity propagation. In addition, we showed that the proposed approach was able to locate important future positions with various qualitative and quantitative evaluations. During the evaluation, we demonstrated that the approach could provide adequate positions in which the field of view of a camera would have to move forward by showing feasible results of the re-targeted video. In Chapter 8, we also proposed a stochastic method of extracting global motions using Gaussian process regression (GPR). Through a stochastic approach that employs mean flow field and variances, we were able to accurately identify the location of importance, even when the input video is a single view and not static and the extracted motions from the video contain a certain amount of noise.

For the motion pattern recognition and anomaly detection tasks, we have shown that the stochastic representation of a motion field using GPR plays a critical role when the input videos have a varying frame rate, and they contain an incomplete motion history at a specific moment. As situations under such videos often occur in real-world online tests, we demonstrated that our proposed approach produced robust results over various types of videos as shown in Chapter 7. Because we were able to generate continuous vector and certainty fields across a spatio-temporally normalized domain, the proposed framework provides instantaneous predictions for pattern recognition and anomalous event detection at each time step of online testing, shown in Section 7.7.

Through various applications and evaluations, we have shown that our hypothesis, which states that motion prediction from spatio-temporal SDI/SDA can effectively solve the sparse data problem in dynamic scene analysis from videos, is feasible and we provide solutions for many dynamic scenes with spatio-temporally sparse data problems. While experiments have shown that each application has some limitations, we have also provided answers to these limitations with possible alternatives. Future directions of study could focus on resolving similar problems of sparsity in other types

of domains.

CHAPTER X

PAPERS RELATED TO THIS THESIS BY THE AUTHOR

1. *Augmenting Aerial Earth Maps with Dynamic Information*, IEEE/ACM International Symposium on Mixed and Augmented Reality, (ISMAR 2009) [69].

2. *Motion Field to Predict Play Evolution in Dynamic Sport Scenes*, IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2010) [67].

3. *Player Localization using Multple Static Cameras for Sports Visualization*, IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2010) [105].

4. *Augmenting Aerial Earth Maps with Dynamic Information from Videos*, Virtual Reality Journal, Springer [70].

5. *Gaussian Process Regression Flow for Analysis of Motion Trajectories*, IEEE International Conference on Computer Vision, (ICCV2011) [68]

6. *Analysis of Motion Fields on the Ground Plane to Predict Play Evolution*, 2011 (under review)

7. *Detecting Regions of Interest in Dynamic Scenes with Camera Motions*, 2011 (under review)

REFERENCES

[1] Agarwal, A., Jawahar, C., and Narayanan, P., "A survey of planar homography estimation techniques," *Technical Report:International Institute of Information Technology*, 2005.

[2] Agarwal, S., Furukawa, Y., Snavely, N., Curless, B., Seitz, S. M., and Szeliski, R., "Reconstructing rome," *Computer*, vol. 43, pp. 40–47, 2010.

[3] Agarwala, A., Zheng, K. C., Pal, C., Agrawala, M., Cohen, M., Curless, B., Salesin, D., and Szeliski, R., "Panoramic video textures," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, (New York, NY, USA), pp. 821–827, ACM, 2005.

[4] Ali, S. and Shah, M., "Floor fields for tracking in high density crowd scenes," in *Proc' of ECCV*, pp. 1–14, 2008.

[5] Ali, S. and Shah, M., "Floor fields for tracking in high density crowd scenes," in *Proc' of the 10th European Conference on Computer Vision: Part II*, pp. 1–14, 2008.

[6] Altun, Y., Hofmann, T., and Smola, A. J., "Gaussian process classification for segmenting and annotating sequences," in *Proc' of ICML*, pp. 4–9, 2004.

[7] Andrew, A. M., "Behavior-based robotics by ronald c. arkin, with a foreword by michael arbib, intelligent robots and autonomous agents series," *Robotica*, vol. 17, no. 2, pp. 229–235, 1999.

[8] Anjyo, K. and Lewis, J., "RBF interpolation and gaussian process regression through an RKHS formulation," *Journal of Math for Industry*, vol. 3, no. 6, pp. 63–71, 2011.

[9] APIDIS, ""http://www.apidis.org/dataset/"," *Autonomous Production of Images Based on Distributed and Intelligent Sensing*, "WWW".

[10] Ariki, Y., Kubota, S., and Kumano, M., "Automatic production system of soccer sports video by digital camera work based on situation recognition," in *ISM*, pp. 851–860, 2006.

[11] Arkin, R. C., *Behavior-Based Robotics*. The MIT. Press, 1998.

[12] Aurenhammer, F., "Voronoi diagrams, A survey of a fundamental geometric data structure," in *ACM Computing Surveys*, pp. 345–405, ACM, 1991.

[13] Ballan, L., Brostow, G. J., Puwein, J., and Pollefeys, M., "Unstructured video-based rendering: Interactive exploration of casually captured videos," vol. 29, p. to appear, July 2010.

[14] Ballan, L., Brostow, G. J., Puwein, J., and Pollefeys, M., "Unstructured video-based rendering: Interactive exploration of casually captured videos," pp. 1–11, July 2010.

[15] Beardsley, P. A., Zisserman, A., and Murray, D. W., "Sequential updating of projective and affine structure from motion," *Int. J. Comput. Vision*, vol. 23, no. 3, pp. 235–259, 1997.

[16] Birkhoff, G., "Piecewise polynomial interpolation and approximation," in *Approximation of functions*, Elsevier, Amsterdam, 1965.

[17] Bouguet, J.-Y., "Pyramidal implementation of the lucas kanade feature tracker," in *Intel Corporation*, 2003.

[18] Buehler, C., Bosse, M., McMillan, L., Gortler, S., and Cohen, M., "Unstructured lumigraph rendering," in *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 425–432, ACM, 2001.

[19] Buhmann, M. D. and Ablowitz, M. J., *Radial Basis Functions : Theory and Implementations.* Cambridge University., 2003.

[20] Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., and Evans, T. R., "Reconstruction and representation of 3d objects with radial basis functions," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, (New York, NY, USA), pp. 67–76, ACM, 2001.

[21] Carr, J. C., Fright, W. R., and Beatson, R. K., "Surface interpolation with radial basis functions for medical imaging," *IEEE Transactions on Medical Imaging*, vol. 16, pp. 96–107, 1997.

[22] Caselles, V., Kimmel, R., and Sapiro, G., "Geodesic active contours," *Int. J. Comput. Vision*, vol. 22, no. 1, pp. 61–79, 1997.

[23] Chan, T. F. and Shen, J. J., "Variational image inpainting," *Communications on Pure and Applied Mathematics*, vol. 58, no. 5, pp. 579–619, 2005.

[24] Chandola, V., Banerjee, A., and Kumar, V., "Anomaly detection: A survey," *ACM Comput. Surv.*, pp. 1–15, 2009.

[25] Chen, F. and Vleeschouwer, C. D., "Distributed video acquisition and annotation for sport-event summarization," in *Networked and Electronic Media Summit*, pp. 1–6, 2008.

[26] Cheng, Y., "Mean shift, mode seeking, and clustering," *IEEE Trans. PAMI*, vol. 17, no. 8, pp. 790–799, 1995.

[27] Chu, W. and Ghahramani, Z., "Preference learning with gaussian processes," in *Proc' of the ICML*, 2005.

[28] Cohen, I. and Herlin, I., "Optical flow and phase portrait methods for environmental satellite image sequences," in *ECCV*, pp. 141–150, 1996.

[29] Comaniciu, D. and Meer, P., "Mean shift analysis and applications," in *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, (Washington, DC, USA), p. 1197, IEEE Computer Society, 1999.

[30] Corpetti, T., Mémin, E., and Pérez, P., "Extraction of singular points from dense motion fields: An analytic approach," *J. Math. Imaging Vis.*, vol. 19, no. 3, pp. 175–198, 2003.

[31] Criminisi, A., Reid, I., and Zisserman, A., "Single view metrology," *IJCV*, vol. 40, pp. 123–148, 1999.

[32] de Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.-P., and Thrun, S., "Performance capture from sparse multi-view video," in *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, (New York, NY, USA), pp. 1–10, ACM, 2008.

[33] Debevec, P. E., Taylor, C. J., and Malik, J., "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *Proc SIGGRAPH '96*, pp. 11–20, 1996.

[34] Duchon, J., *Splines minimizing rotation-invariant semi-norms in Sobolev spaces*. Lecture Notes in Mathematics, Springer-Verlag, 1977.

[35] Efros, A. A., Berg, E. C., Mori, G., and Malik, J., "Recognizing action at a distance," in *ICCV03*, pp. 726–733, 2003.

[36] Einarsson, P., Chabert, C.-F., Jones, A., Ma, W.-C., Lamond, B., Hawkins, T., Bolas, M., Sylwan, S., and Debevec, P., "Relighting human locomotion with flowed reflectance fields," in *Eurographics Symposium on Rendering, 2006*, pp. 31–42, 2006.

[37] ElectronicArts, ""NHL Hockey '09"," 2009.

[38] Ellis, D., Sommerlade, E., and Reid, I., "Modelling pedestrian trajectories with gaussian processes," in *International Workshop on Visual Surveillance*, pp. 1229–1234, 2009.

[39] Eshel, R. and Moses, Y., "Homography based multiple camera detection and tracking of people in a dense crowd," in *CVPR*, pp. 1–8, 2008.

[40] Eyevision, ""http://www.ri.cmu.edu/events/sb35/tksuperbowl.html"," "WWW".

[41] Frey, B. and MacKay, D., "A revolution: Belief propagation in graphs with cycles," in *Neural Information Processing Systems*, pp. 479–485, 1998.

[42] Funge, J., Tu, X., and Terzopoulos, D., "Cognitive modeling: knowledge, reasoning and planning for intelligent characters," in *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 29–38, ACM Press/Addison-Wesley Publishing Co., 1999.

[43] Girgensohn, A., Kimber, D., Vaughan, J., Yang, T., Shipman, F., Turner, T., Rieffel, E., Wilcox, L., Chen, F., and Dunnigan, T., "Dots: support for effective video surveillance," in *ACM MULTIMEDIA '07*, (New York, NY, USA), pp. 423–432, ACM, 2007.

[44] Goldenstein, S., Karavelas, M. I., Metaxas, D. N., Guibas, L. J., Aaron, E., and Goswami, A., "Scalable nonlinear dynamical systems for agent steering and crowd simulation," *Computers & Graphics*, vol. 25, no. 6, pp. 983–998, 2001.

[45] Google Inc., ""Google Street View http://maps.google.com/help/maps/streetview/"," "WWW".

[46] Goovaerts, P., "Geostatistical analysis of disease data: estimation of cancer mortality risk from empirical frequencies using poisson kriging," *International Journal of Health Geographics*, vol. 4, no. 31, pp. 63–71, 2005.

[47] Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F., "The lumigraph," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 43–54, ACM, 1996.

[48] Grundmann, M., Kwatra, V., Han, M., and Essa, I., "Efficient hierarchical graph based video segmentation," *IEEE CVPR*, 2010.

[49] Harris, M. J., "Real-time cloud simulation and rendering," in *ACM SIGGRAPH 2005 Courses*, (New York, NY, USA), p. 222, 2005.

[50] Hartley, R. and Zisserman, A., *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, 2000.

[51] Hess, R. and Fern, A., "Improved video registration using non-distinctive local image features," in *2007 IEEE CVPR*, pp. 18–23, IEEE Computer Society, 2007.

[52] Hornby, P. and Horowitz, F., "Closely related alternatives to kriging interpolation," in *Unpublished manuscript*, pp. 1–5, 1996.

[53] Horry, Y., Anjyo, K.-I., and Arai, K., "Tour into the picture: using a spidery mesh interface to make animation from a single image," in *Proc. ACM SIGGRAPH*, (New York, NY, USA), pp. 225–232, 1997.

[54] Hu, W., Xiao, X., Fu, Z., Xie, D., Tan, T., and Maybank, S., "A system for learning statistical motion patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, pp. 1450–1464, 2006.

[55] Isard, M. and Blake, A., "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[56] J.-P. Costa, L. P. and Thierry, E., "A comparison between kriging and radial basis function networks for nonlinear prediction," in *IEEE EURASIP Workshop on Nonlinear Signal and Image Processing*, pp. 1–5, IEEE, 1999.

[57] Julier, S. J. and Uhlmann, J. K., "A new extension of the kalman filter to nonlinear systems," pp. 182–193, 1997.

[58] Kalaitzis, A., "Image inpainting with gaussian processes," in *Thesis*, 2009.

[59] Kalman, R. E., "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[60] Kanade, T. and Morris, D. D., "Factorization methods for structure from motion," *Philosophical Transactions of the Royal Society of London, Series A*, vol. 356, no. 1, pp. 1153–1173, 1998.

[61] Keogh, E. J. and Pazzani, M. J., "Derivative Dynamic Time Warping," in *SIAM SDM'01*, 2001.

[62] Khalid, S. and Naftel, A., "Classifying spatiotemporal object trajectories using unsupervised learning of basis function coefficients," in *Proc' of the third ACM international workshop on VSSN*, pp. 45–52, 2005.

[63] Khan, S. and Shah, M., "Object based segmentation of video using color, motion and spatial information," in *CVPR01*, pp. 746–751, 2001.

[64] Khan, S. M. and Shah, M., "A multiview approach to tracking people in crowded scenes using a planar homography constraint," in *ECCV*, 2006.

[65] Khan, Z., Balch, T., and Dellaert, F., "Mcmc-based particle filtering for tracking a variable number of interacting targets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1805–1918, 2005.

[66] Kim, K., Grundmann, M., Shamir, A., Matthews, I., Hodgins, J., and Essa, I., "Motion field to predict play evolution," *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 2010.

[67] Kim, K., Grundmann, M., Shamir, A., Matthews, I., Hodgins, J., and Essa, I., "Motion field to predict play evolution in dynamic sport scenes," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010*, pp. 840–847, 2010.

[68] Kim, K., Lee, D., and Essa, I., "Gaussian process regression flow for analysis of motion trajectories," in *In Proceeding of 2011 IEEE International Conference on Computer Vision*, 2011.

[69] Kim, K., Oh, S., Lee, J., and Essa, I., "Augmenting aerial earth maps with dynamic information," in *IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR*, 2009.

[70] Kim, K., Oh, S., Lee, J., and Essa, I., "Augmenting aerial earth maps with dynamic information from videos," *Virtual Reality Journal, Springer*, 2011.

[71] Kim, K. and Davis, L., "Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering," in *ECCV*, pp. 98–109, 2006.

[72] Koller-Meier, E. and Ade, F., "Tracking multiple objects using the condensation algorithm," *Robotics and Autonomous Systems*, vol. 34, no. 2-3, pp. 93–105, 2001.

[73] Kosecka, J. and Zhang, W., "Video compass," in *Proceedings of ECCV*, (London, UK), pp. 476–490, Springer-Verlag, 2002.

[74] Lab, L., ""second life - http://www.liberovision.com/"," "WWW".

[75] Laurentini, A., "The visual hull concept for silhouette-based image understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 2, pp. 150–162, 1994.

[76] Lazebnik, S., Furukawa, Y., and Ponce, J., "Projective visual hulls," *Int. J. Comput. Vision*, vol. 74, no. 2, pp. 137–165, 2007.

[77] Lee, D.-S., "Effective gaussian mixture learning for video background subtraction," *IEEE PAMI*, vol. 27, no. 5, 2005.

[78] Lee, S.-I., Lee, H., Abbeel, P., and Ng, A. Y., "Efficient L1 regularized logistic regression," in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

[79] Levin, A. and Weiss, Y., "Learning to combine bottom-up and top-down segmentation," *Int. J. Comput. Vision*, vol. 81, no. 1, pp. 105–118, 2009.

[80] Levoy, M. and Hanrahan, P., "Light field rendering," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 31–42, ACM, 1996.

[81] Lewis, J. P., ed., *Algorithms for Solid Noise Synthesis*, 1989.

[82] Lewis, J. and Anjyo, K., "Scattered data interpolation for computer graphics," in *ACM Siggraph Asia 2010, Course*, pp. 1–84, ACM, 2010.

[83] Li, L., Huang, W., Gu, I. Y. H., and Tian, Q., "Foreground object detection from videos containing complex background," in *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, (New York, NY, USA), pp. 2–10, ACM, 2003.

[84] Li, X., Hu, W., and Hu, W., "A coarse-to-fine strategy for vehicle motion trajectory clustering," in *Proc' of the ICPR '06*, pp. 591–594, 2006.

[85] LiberoVision, ""http://www.liberovision.com/"," "WWW".

[86] Lin, J., Keogh, E., Lonardi, S., and Chiu, B., "A symbolic representation of time series, with implications for streaming algorithms," in *Proc' s of ACM SIGMOD workshop on data mining and knowledge discovery*, 2003.

[87] Longuet-Higgins, H. C., "A computer algorithm for reconstructing a scene from two projections," pp. 61–62, 1987.

[88] Man, P., "Generating and real-time rendering of clouds," in *Central European Seminar on Computer Graphics*, pp. 1–9, 2006.

[89] Meijering, E., "A chronology of interpolation: from ancient astronomy to modern signal and image processing," in *Proceeding of IEEE*, pp. 319–342, IEEE, 2001.

[90] MocapDatabase, C., ""CMU Graphics Lab Motion Capture Database http://mocap.cs.cmu.edu/"," "WWW".

[91] Morris, B. and Trivedi, M., "Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes," in *Proc'IEEE CVPR '09*, pp. 1–8, 2009.

[92] Morvan, Y. and O'sullivan, C., "A perceptual approach to trimming and tuning unstructured lumigraphs," *ACM Trans. Appl. Percept.*, vol. 5, no. 4, pp. 1–24, 2009.

[93] Neumann, U., Pintaric, T., and Rizzo, A., "Immersive panoramic video," in *MULTIMEDIA '00: Proceedings of the eighth ACM international conference on Multimedia*, (New York, NY, USA), pp. 493–494, ACM, 2000.

[94] Neumann, U., You, S., Hu, J., Jiang, B., and Lee, J., "Augmented virtual environments (ave): for visualization of dynamic imagery," in *IEEE Virtual Reality03*, pp. 61–67, 2003.

[95] Nocedal, J. and Wright, S., "Numerical Optimization, Series in Operations Research and Financial Engineering," 2006.

[96] Oh, S. and Hoogs, A., "Unsupervised learning of activities in video using scene context," in *Proc' of ICPR*, pp. 3579–3582, 2010.

[97] Oliver, N. M., Rosario, B., and Pentland, A. P., "A bayesian computer vision system for modeling human interactions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 831–843, 2000.

[98] PajamaChannel, ""http://www.pajamachannels.com/"."

[99] Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[100] Pérez, P., Gangnet, M., and Blake, A., "Poisson image editing," in *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, (New York, NY, USA), pp. 313–318, ACM, 2003.

[101] Perlin, K., "An image synthesizer," *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 287–296, 1985.

[102] Piciarelli, C. and Foresti, G. L., "On-line trajectory clustering for anomalous events detection," *Pattern Recogn. Lett.*, vol. 27, pp. 1835–1842, 2006.

[103] Rabaud, V. and Belongie, S., "Counting crowded moving objects," in *CVPR '06: Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 17–22, IEEE Computer Society, 2006.

[104] Rabiner, L. and Juang, B.-H., *Fundamentals of speech recognition*. 1993.

[105] Raffay Hamid, Ram Krishan Kumar, M. G. K. K. I. E. J. H., "Player localization using multiple static cameras for sports visualization," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010*, 2010.

[106] Ramanan, D. and Forsyth, D. A., "Automatic annotation of everyday movements," in *in NIPS*, MIT Press, 2003.

[107] Rao, A. R. and Jain, R. C., "Computerized flow field analysis: Oriented texture fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 7, pp. 693–709, 1992.

[108] Rasmussen, C., "Gaussian processes for machine learning," in *Book, MIT-Press*, MIT-Press, 2006.

[109] Rasmussen, C., "Gaussian processes in machine learning," *Advanced Lectures on Machine Learning*.

[110] Raykar, V. C. and Duraiswami, R., "Fast large scale gaussian process regression using approximate matrix-vector products," 2007.

[111] Reynolds, C. W., "Flocks, herds and schools: A distributed behavioral model," in *ACM SIGGRAPH 1987*, (New York, NY, USA), pp. 25–34, ACM Press, 1987.

[112] Reynolds, C. W., "Steering behaviors for autonomous characters," in *GDC '99: Proceedings of Game Developers Conference*, pp. 768–782, Miller Freeman Game Group, 1999.

[113] Sawhney, H. S., Arpa, A., Kumar, R., Samarasekera, S., Aggarwal, M., Hsu, S., Nister, D., and Hanna, K., "Video flashlights: real time rendering of multiple videos for immersive model visualization," in *13th Eurographics workshop on Rendering*, pp. 157–168, Eurographics Association, 2002.

[114] Schindler, G., Krishnamurthy, P., and Dellaert, F., "Line-based structure from motion for urban environments," in *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, (Washington, DC, USA), pp. 846–853, IEEE Computer Society, 2006.

[115] Schödl, A., Szeliski, R., Salesin, D. H., and Essa, I., "Video textures," in *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 489–498, ACM Press/Addison-Wesley Publishing Co., 2000.

[116] Sebe, I. O., Hu, J., You, S., and Neumann, U., "3d video surveillance with augmented virtual environments," in *IWVS '03: First ACM SIGMM international workshop on Video surveillance*, (New York, NY, USA), pp. 107–112, ACM, 2003.

[117] Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R., "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *IEEE CVPR '06*, pp. 519–528, 2006.

[118] Seitz, S. M. and Dyer, C. R., "View morphing," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 21–30, ACM, 1996.

[119] Shi, J. and Malik, J., "Normalized cuts and image segmentation," in *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, (Washington, DC, USA), p. 731, IEEE Computer Society, 1997.

[120] Shi, J. and Tomasi, C., "Good features to track," in *Proceedings of IEEE CVPR*, pp. 593–600, IEEE Computer Society, 1994.

[121] Smart, J., Cascio, J., and Paffendorf, J., "Metaverse roadmap : Pathways to the 3d web," *Metaverse : A Cross-Industry Public Foresight Project*, 2007.

[122] Snavely, N., Seitz, S. M., and Szeliski, R., "Photo tourism: Exploring photo collections in 3d," in *SIGGRAPH Conference Proceedings*, (New York, NY, USA), pp. 835–846, ACM Press, 2006.

[123] Snavely, N., Seitz, S. M., and Szeliski, R., "Modeling the world from Internet photo collections," *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, 2008.

[124] Spokas, K., Graff, C., Morcet, M., and Aran, C., "Implications of the spatial variability of landfill emission rates on geospatial analyses," *International Landfill Research Symposium*, vol. 23, no. 7, pp. 599–607, 2003.

[125] Stauffer, C. and Grimson, W. E. L., "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, 2000.

[126] Szenberg, F., Carvalho, P. C. P., and Gattass, M., "Automatic camera calibration for image sequences of a football match," in *ICAPR*, (London, UK), 2001.

[127] Tan, P.-N., Steinbach, M., and Kumar, V., *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[128] Tokai, S., Yumiba, R., Wu, X., and Matsuyama, T., "Dynamic wide-area scene visualization using an active camera," *Syst. Comput. Japan*, vol. 37, no. 8, pp. 101–112, 2006.

[129] Tong, Y., Lombeyda, S., Hirani, A. N., and Desbrun, M., "Discrete multiscale vector field decomposition," *ACM ToG*, vol. 22, no. 3, pp. 445–452, 2003.

[130] Treuille, A., Cooper, S., and Popović, Z., "Continuum crowds," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, (New York, NY, USA), pp. 1160–1168, ACM, 2006.

[131] Turk, G. and O'Brien, J. F., "Shape transformation using variational implicit functions," in *SIGGRAPH '99*, (New York, NY, USA), pp. 335–342, 1999.

[132] Urtasun, R., Fleet, D. J., and Fua, P., "3d people tracking with gaussian process dynamical models," in *Proc' of CVPR*, pp. 238–245, 2006.

[133] US AIR FORCE, "CLIF https://www.sdms.afrl.af.mil/,"

[134] Vedula, S., Baker, S., and Kanade, T., "Spatio-temporal view interpolation," in *Proceedings of the 13th ACM Eurographics Workshop on Rendering*, June 2002.

[135] Veenman, C. J., Reinders, M. J. T., and Backer, E., "Resolving motion correspondence for densely moving points," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 54–72, 2001.

[136] Verri, A. and Poggio, T., "Motion field and optical flow: Qualitative properties," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 5, pp. 490–498, 1989.

[137] Vlachos, M., Gunopoulos, D., and Kollios, G., "Discovering similar multidimensional trajectories," in *Proc' ICDE*, pp. 673–680, 2002.

[138] Vlasic, D., Baran, I., Matusik, W., and Popović, J., "Articulated mesh animation from multi-view silhouettes," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–9, 2008.

[139] Wahba, G., "Spline interpolation and smoothing on the sphere," *SIAM Journal of Scientific Computing*, vol. 2, no. 1, pp. 1064–8275, 1979.

[140] Wang, J. M., Fleet, D. J., and Hertzmann, A., "Gaussian process dynamical models for human motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, pp. 283–298, 2008.

[141] Wang, J. Y. and Adelson, E. H., "Representing moving images with layers," *IEEE Trans. on Image Proc.*, vol. 3, pp. 625–638, 1994.

[142] Wang, N., "Realistic and fast cloud rendering," *Journal of Graphics Tools*, vol. 9, no. 3, pp. 21–40, 2004.

[143] Wang, Y., Krum, D. M., Coelho, E. M., and Bowman, D. A., "Contextualized videos: Combining videos with environment models to support situational understanding," vol. 13, pp. 1568–1575, 2007.

[144] Wood, D. M., Ball, K., Lyon, D., Norris, C., and Raab, C., "A report on the surveillance society," *Surveillance Studies Network*, 2006.

[145] WWW, "http://www.justin.tv/adamsblock,"

[146] Xiao, J., Yang, C., Han, F., and Cheng, H., "Multimodal technologies for perception of humans," in *Vehicle and Person Tracking in Aerial Videos*, pp. 203–214, 2008.

[147] Yankov, D., Keogh, E., Medina, J., Chiu, B., and Zordan, V., "Detecting time series motifs under uniform scaling," in *Proc' of ACM SIGKDD '07*, pp. 844–853, 2007.

[148] Ye, Y. and Liu, C. K., "Synthesis of responsive motion using a dynamic model," *Comput. Graph. Forum*, pp. 555–562, 2010.

[149] Zach, C., Pock, T., and Bischof, H., "A globally optimal algorithm for robust tv-l1 range image integration," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1–8, IEEE, 2007.

[150] Zapletal, J., Vaněček, P., and Skala, V., "Rbf-based image restoration utilising auxiliary points," in *Proceedings of the 2009 Computer Graphics International Conference*, CGI '09, (New York, NY, USA), pp. 39–43, ACM, 2009.

[151] Zhang, Z., Huang, K., and Tan, T., "Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes," in *Proc' of ICPR '06*, pp. 1135–1138, 2006.

[152] Zotti, G. and Groller, M. E., "A sky dome visualisation for identification of astronomical orientations," in *INFOVIS '05*, (Washington, DC, USA), p. 2, IEEE Computer Society, 2005.

VITA

Kihwan Kim was born in Seoul, South Korea, on March 7, 1975. After completing his work at Kyungmoon high school, Seoul, South Korea in 1994, he went to Yonsei University in Seoul, from 1994 to 2001. He graduated with a Bachelor of Science degree in Electrical Engineering in 2001. During his undergraduate study, he performed his military service in the Republic of Korea Air Force in Suwon, South Korea, from 1996 to 1998, as an avionic engineer.

From February 2001 to June 2005, he worked for Samsung SDS, Information and Technology R&D center in Seoul, South Korea, as an advisory engineer. He also worked for Digital Solution Center in Samsung Electronics as an ubiquitous task force.

He moved to Atlanta, GA, USA in August 2005, and earned Master of Science degree in computer science from Georgia Institute of Technology in May 2010, and defended his dissertation for the degree of Doctor of Philosophy in computer science from Georgia Institute of Technology in December 2011. During his doctoral study, he joined a research group in Disney Research Pittsburgh from January 2009 to August 2009. In January 2012, he joined a mobile visual computing research group in NVIDIA Research as a research scientist.

He works in the areas of computer vision and computer graphics with potential impact on image/video analysis (e.g, computational photography, scene analysis/visualization, and image/video based modeling and rendering). Specifically, he is interested in the analysis, interpretation, and synthesis of videos with the goals of visualizing dynamic scenes in various scales of outdoor environment.